

**Learning Bayesian network
equivalence classes using
ant colony optimisation**

Rónán Daly



Doctor of Philosophy
Centre for Intelligent Systems and their Applications
School of Informatics
University of Edinburgh
2008

Abstract

Bayesian networks have become an indispensable tool in the modelling of uncertain knowledge. Conceptually, they consist of two parts: a directed acyclic graph called the structure, and conditional probability distributions attached to each node known as the parameters. As a result of their expressiveness, understandability and rigorous mathematical basis, Bayesian networks have become one of the first methods investigated, when faced with an uncertain problem domain. However, a recurring problem persists in specifying a Bayesian network. Both the structure and parameters can be difficult for experts to conceive, especially if their knowledge is tacit.

To counteract these problems, research has been ongoing, on learning both the structure and parameters of Bayesian networks from data. Whilst there are simple methods for learning the parameters, learning the structure has proved harder. Part of this stems from the NP-hardness of the problem and the super-exponential space of possible structures. To help solve this task, this thesis seeks to employ a relatively new technique, that has had much success in tackling NP-hard problems. This technique is called ant colony optimisation. Ant colony optimisation is a metaheuristic based on the behaviour of ants acting together in a colony. It uses the stochastic activity of artificial ants to find good solutions to combinatorial optimisation problems. In the current work, this method is applied to the problem of searching through the space of equivalence classes of Bayesian networks, in order to find a good match against a set of data. The system uses operators that evaluate potential modifications to a current state. Each of the modifications is scored and the results used to inform the search. In order to facilitate these steps, other techniques are also devised, to speed up the learning process. The techniques include faster versions of tests needed whilst performing a search and caching of the test results.

The techniques are tested by sampling data from gold standard networks and learning structures from this sampled data. These structures are analysed using various goodness-of-fit measures to see how well the algorithms perform. The measures include structural similarity metrics and Bayesian scoring metrics. The results are compared in depth against systems that also use ant colony optimisation and other methods, including evolutionary programming and greedy heuristics. Also, comparisons are made to well known state-of-the-art algorithms and a study performed on a real-life data set. The results show favourable performance compared to the other methods and on modelling the real-life data.

Acknowledgements

Firstly, thanks must go to my principal supervisor, Professor Qiang Shen of Aberystwyth University, for his guidance, invaluable advice and motivational ability. His professionalism and ability to provide quick feedback on my work have helped immensely in being able to finish this study.

Thanks also to my second supervisor, Doctor Stuart Aitken of the University of Edinburgh, who has been a great help in the day-to-day problems and provided valuable feedback on all aspects of this work.

Many thanks go to my examiners, Doctors Alex Freitas of the University of Kent and Gillian Hayes of the University of Edinburgh. Their attention to detail ensured that this thesis is much better than originally put forward.

I must especially acknowledge the help of my parents in being able to pursue this course: my father John and mother Máiréad. Without their continued love and support I would not have been able to start or continue this work. For this I will ever be grateful.

Thanks to my sister Bláithín and brother-in-law Ed, whose kind support is always appreciated.

Special thanks of course go to my other family in Philadelphia, Mariclare and Tim, who always made sure I never lost sight of the lighter side of life.

Finally, thanks to my partner Ash. She has stuck with me through all the travails of this thesis and made sure I got there in the end. Her sense of humour, kindness and patience were second to none and ensured that I made it this far.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(Rónán Daly)

To my dearest Ash, without whom none of this would have been possible

Contents

1	Introduction	1
1.1	Motivation	1
1.1.1	Using Bayesian Networks to Encode Knowledge	2
1.1.2	Using Machine Learning Techniques to Construct Bayesian Networks	2
1.1.3	Using Ant Colony Optimisation to Learn Bayesian Networks	3
1.2	Thesis Objectives	4
1.3	Methodology	5
1.3.1	Searching the Space of Equivalence Classes Using a Novel ACO Algorithm	5
1.3.2	Speeding up Search in the Space of Equivalence Classes	6
1.3.3	Testing the Hypothesised Methods	6
1.4	Preview of Results	6
1.5	Dissertation Structure	8
2	A Review of the Literature on Learning Bayesian Networks	11
2.1	Introduction to Bayesian Networks	12
2.1.1	Preliminaries	12
2.1.2	Bayesian Networks	14
2.1.3	Markov Equivalent Structures	17
2.1.4	Special Types of Bayesian Networks	19
2.2	Inference in Bayesian Networks	21
2.2.1	Introduction to Inference	22
2.2.2	Trees and Polytrees	22
2.2.3	Multiply Connected Networks	24
2.2.4	Clustering	24
2.2.5	Variable Elimination and Arc Reversal	25

2.2.6	Conditioning	25
2.2.7	Symbolic Probabilistic Inference	25
2.2.8	Polynomial Compilation	26
2.2.9	Monte-Carlo Methods	26
2.2.10	Other Approximate Inference	27
2.2.11	Inference in Dynamic Bayesian Networks	29
2.2.12	Causal Independence Networks	29
2.3	Learning Bayesian Network Parameters	29
2.3.1	Multinomial Data	30
2.3.2	Continuous Variables	31
2.3.3	Missing Data/Hidden Variables	31
2.3.4	Miscellaneous Techniques	33
2.4	Learning Bayesian Network Structures	34
2.4.1	Learning Theory and Learning Complexity	34
2.4.2	Trees	35
2.4.3	Polytrees	36
2.4.4	Heuristic Algorithms	36
2.4.5	Searching Through the Space of Equivalence Classes	40
2.4.6	Scoring Functions	42
2.4.7	Finding Structure using Conditional Independencies	45
2.4.8	Hybrid Search Strategies	48
2.4.9	Searching over Orderings	49
2.4.10	Dynamic Programming	50
2.4.11	Missing Data and Partially Observed Data	50
2.4.12	Model Averaging	52
2.4.13	Dynamic Bayesian Networks	53
2.4.14	Parallel Learning	54
2.4.15	On-line Learning	55
2.4.16	Incorporating Prior Knowledge	55
2.4.17	Large Domains	56
2.4.18	Continuous Variables	57
2.4.19	Robustness	58
2.4.20	Acceleration Techniques	58
2.4.21	Miscellaneous Techniques	59
2.5	Applications	60

2.6	Comparison of Techniques and Summary	62
2.6.1	Tree and Polytrees	63
2.6.2	Heuristic Search	63
2.6.3	Conditional Independencies	64
2.6.4	Dynamic Programming	64
2.6.5	Summary	64
3	A Review of the Literature on Ant Colony Optimisation	67
3.1	Ant Colony Optimisation	67
3.1.1	Swarm Intelligence	68
3.1.2	Ant Colony Optimisation	69
3.2	The ACO Metaheuristic	70
3.3	Example – The Travelling Salesman Problem	72
3.4	ACO in Machine Learning Problems	75
3.5	ACO in Learning Bayesian Network Structures	77
3.5.1	ACO-K2SN	78
3.5.2	ACO-B	79
3.5.3	Performance Comparison	80
3.6	Summary	80
4	Using ACO in the Learning of Bayesian Network Equivalence Classes	83
4.1	Score and Search Based Methods of Learning Bayesian Network Structures	84
4.1.1	Techniques for Searching Through Equivalence Classes	87
4.1.2	Advantages of Searching in E-space	89
4.2	Accelerating the Learning Process	89
4.2.1	Reducing the Number of Checked Nodes	90
4.2.2	Caching	95
4.3	Using Ant Colony Optimisation in Learning an Equivalence Class	99
4.3.1	Relation of ACO-E to the ACO Metaheuristic	100
4.3.2	Description of ACO-E	106
4.3.3	Implementation Issues	108
4.4	Summary	109
5	Experimental Methodology	113
5.1	Standard Bayesian Networks	114
5.1.1	Six Gold-Standard Networks	114

5.1.2	Sampling Data from a Network	115
5.2	Accelerating the Learning Process	116
5.2.1	Experimental Design	116
5.2.2	Evaluation Criteria	121
5.3	Using Ant Colony Optimisation in Learning an Equivalence Class . . .	122
5.3.1	Experimental Design	122
5.3.2	Evaluation Criteria	127
5.4	Summary	129
6	Results and Discussion	131
6.1	Accelerating the Learning Process	131
6.1.1	Results	132
6.1.2	Discussion	132
6.2	Using Ant Colony Optimisation in Learning an Equivalence Class . . .	142
6.2.1	Results	142
6.2.2	Discussion	159
6.3	Summary	172
7	Modelling the Circadian Clock of <i>Arabidopsis Thaliana</i>	175
7.1	Background	176
7.1.1	Data from <i>Arabidopsis thaliana</i> Experiments	177
7.2	Modelling Gene Expression Data using DBNs	178
7.2.1	Using ACO-E to Learn a Dynamic Bayesian Network	179
7.3	Experimental Methodology	182
7.3.1	Preprocessing the Data	182
7.3.2	Prior Knowledge	184
7.3.3	Testing Methodology	185
7.3.4	Evaluation Criteria	185
7.4	Results and Discussion	188
7.4.1	Discussion of Experimental Results	189
7.5	Summary	199
8	Conclusions and Future Work	201
8.1	Conclusions	202
8.1.1	Accelerating the Search Through the Space of Equivalence Class- es of Bayesian Network Structures	202

8.1.2	Using Ant Colony Optimisation in Learning Equivalence Classes of Bayesian Network Structures	202
8.1.3	Building a Dynamic Bayesian Network using ACO-E to Model the Circadian Clock of <i>Arabidopsis Thaliana</i>	204
8.2	Future Work	205
8.2.1	Speeding up ACO-E and Similar Algorithms	205
8.2.2	Extending ACO-E to Increase Performance and Scalability . . .	206
8.2.3	Applying ACO-E to Real-Life Data	207
8.3	Summary	208
A	Experimental Bayesian Networks	209
	Bibliography	217

List of Figures

2.1	A directed acyclic graph	15
2.2	The skeleton of the DAG in Figure 2.1	17
2.3	V-Structures	18
2.4	A partially directed acyclic graph	18
2.5	A PDAG for which there exists no consistent extension	19
2.6	Dynamic Bayesian network structures	20
2.7	The ASIA Bayesian network structure	23
2.8	Inference by message passing	23
4.1	Validity of moves not changing	95
4.2	Adding edges can affect nodes far away	98
4.3	An example partial construction graph for the three node network shown in Figure 4.4	102
4.4	A three node network used for illustration by Figure 4.3	103
5.1	Distribution of nodes across in-degree – Alarm	117
5.2	Distribution of nodes across in-degree – Barley	117
5.3	Distribution of nodes across in-degree – Diabetes	118
5.4	Distribution of nodes across in-degree – HailFinder	118
5.5	Distribution of nodes across in-degree – Mildew	119
5.6	Distribution of nodes across in-degree – Win95pts	119
5.7	Adding v-structures	129
6.1	Comparison of original and new validity checking – Alarm	133
6.2	Comparison of original and new validity checking – Barley	133
6.3	Comparison of original and new validity checking – Diabetes	134
6.4	Comparison of original and new validity checking – HailFinder	134
6.5	Comparison of original and new validity checking – Mildew	135

6.6	Comparison of original and new validity checking – Win95pts	135
6.7	Original and new running times	136
6.8	Speed-up ratio	136
6.9	BDeu scores for metaheuristic algorithm comparison – Alarm	149
6.10	BDeu scores for metaheuristic algorithm comparison – Barley	149
6.11	BDeu scores for metaheuristic algorithm comparison – Diabetes	150
6.12	BDeu scores for metaheuristic algorithm comparison – HailFinder	150
6.13	BDeu scores for metaheuristic algorithm comparison – Mildew	151
6.14	BDeu scores for metaheuristic algorithm comparison – Win95pts	151
6.15	SHD for metaheuristic algorithm comparison – Alarm	152
6.16	SHD for metaheuristic algorithm comparison – Barley	152
6.17	SHD for metaheuristic algorithm comparison – Diabetes	153
6.18	SHD for metaheuristic algorithm comparison – HailFinder	153
6.19	SHD for metaheuristic algorithm comparison – Mildew	154
6.20	SHD for metaheuristic algorithm comparison – Win95pts	154
6.21	ESHD for metaheuristic algorithm comparison – Alarm	155
6.22	ESHD for metaheuristic algorithm comparison – Barley	155
6.23	ESHD for metaheuristic algorithm comparison – Diabetes	156
6.24	ESHD for metaheuristic algorithm comparison – HailFinder	156
6.25	ESHD for metaheuristic algorithm comparison – Mildew	157
6.26	ESHD for metaheuristic algorithm comparison – Win95pts	157
6.27	SHD results across various state-of-the-art algorithms – Alarm	160
6.28	SHD results across various state-of-the-art algorithms – Barley	160
6.29	SHD results across various state-of-the-art algorithms – HailFinder	161
6.30	SHD results across various state-of-the-art algorithms – Mildew	161
7.1	<i>Arabidopsis thaliana</i>	177
7.2	A Bayesian network unrolled to a dynamic Bayesian network	179
7.3	A 4-layer dynamic Bayesian network	180
7.4	Expression level of PRR9 in the AT0029 condition	183
7.5	Expression level derivative of PRR9 in the AT0029 condition	183
7.6	Expert developed DBN for <i>Arabidopsis thaliana</i> circadian clock	186
7.7	ROC for the time independent TPR and FPR for all data	191
7.8	The time independent TPR and FPR as a function of N' for all data	191
7.9	ROC for the time dependent TPR and FPR for all data	192

7.10	The time dependent TPR and FPR as a function of N' for all data	192
7.11	Plots of the positive rates for the AT0029 condition	193
7.12	Plots of the positive rates for the AT0030 condition	194
7.13	Plots of the positive rates for the AT0031b condition	195
7.14	Plots of the positive rates for the AT0032 condition	196
7.15	Plots of the positive rates for the AT0033 condition	197
7.16	Plots of the positive rates for the AT0047 condition	198
A.1	The Alarm Bayesian network	210
A.2	The Barley Bayesian network	211
A.3	The Diabetes Bayesian network	212
A.4	The HailFinder Bayesian network	213
A.5	The Mildew Bayesian network	214
A.6	The Win95pts Bayesian network	215

List of Tables

4.1	Basic modification operators	86
4.2	Validity conditions and change in score for each operator	91
4.3	Validity conditions and set of valid nodes for a node x	92
4.4	The nodes that must be checked for a change at node x	97
5.1	Bayesian network properties	115
5.2	Parameter values for testing acceleration methods	121
5.3	Parameter values for testing ACO-E with experimental condition 1 . . .	124
5.4	Parameter values for testing ACO-E experimental condition 2	126
6.1	Mean and standard deviation of running times at $t = 100$ over 100 runs	132
6.2	t values and p-values for comparing the average run times	139
6.3	r values and p-values for comparing the average run times across iterations	140
6.4	Mean and standard deviation of the BDeu score for ACO-E for each parameter setting (the bold figure is the best for that row)	144
6.5	Mean and standard deviation of the BDeu score for ACO-E for each parameter setting (the bold figure is the best for that row)	145
6.6	Mean and standard deviation of the SHD for ACO-E for each parameter setting (the bold figure is the best for that row)	146
6.7	Mean and standard deviation of the ESHD for ACO-E for each parameter setting (the bold figure is the best for that row)	147
6.8	Mean and standard deviation for metaheuristic algorithms (the bold figure is the best for that row)	148
6.9	SHD mean and standard deviation for state-of-the-art algorithms (the bold figure is the best for that column)	158
6.10	Comparisons of parameter behaviour	164
6.11	p-values for Mann-Whitney U test, 10,000 samples	169
6.12	p-values for Conover's squared ranks test, 10,000 samples	170

6.13	p-values comparing ACO-E against state-of-the-art algorithms	171
7.1	Ratio of light to darkness for each experimental condition	178
7.2	Extra conditions for operators in a dynamic Bayesian network	181
7.3	Area under the ROC curve for both the time independent and time dependent true positive cases	188

List of Algorithms

4.1	UPDATE-CACHE	96
4.2	ACO-E	111
4.3	ANT-E	112
4.4	TOTAL-SCORE	112
5.1	Generating samples from a Bayesian network	116
5.2	The SHD metric	128

Nomenclature

N_x The neighbour set of node x

Π_x The parents of node x

Ξ_x The children of node x

$X_{N|\Xi|\Pi}$ Nodes that can be reached from node x by following neighbours (N), children (Ξ) or parents (Π)

M^{-x} $M \setminus \{x\}$

M^{+x} $M \cup \{x\}$

N_{Π_x} $\bigcup_{\pi \in \Pi_x} N_\pi$

$N_{x,y}$ $N_x \cap N_y$

$\Omega_{x,y}$ $\Pi_x \cap N_y$

Chapter 1

Introduction

BAYESIAN NETWORKS are mathematical objects that can be used to provide a factored representation of a probability distribution. They consist of parameters and a structure that specifies how these parameters are combined to produce the distribution. As a result of their representational power and because the structured part of the network can model causal relations in a manner understandable to humans, they are now seen as a useful mechanism to encode uncertain knowledge.

This thesis will apply the ant colony optimisation technique to the problem of learning Bayesian network structures. As an introduction, this chapter will give an overview of the path followed in achieving this task. Furthermore, a discussion will be given of: the motivation behind the problem, the thesis' objectives, the methodology for achieving the objectives, a concise examination of the results found and an overview of the structure of this dissertation.

1.1 Motivation

This section will examine three of the main motivating points behind this work and seek to answer the questions behind them:

- Why should knowledge be represented as a Bayesian network?
- Why should machine learning techniques be used to construct Bayesian networks?
- Why should ACO be used as a learning technique – in particular for the problem above?

These points will now be discussed.

1.1.1 Using Bayesian Networks to Encode Knowledge

Bayesian networks have become a mainstream tool in the representation of uncertain knowledge. There are many reasons for this. Perhaps the most influential is that the structure of Bayesian networks has an intuitive feel, as humans can understand what is being represented fairly easily. However, an easily understandable structure is not enough. Underlying Bayesian networks is the safety net of probability, which guarantees that well understood and useful mathematical properties apply. Bayesian networks can be seen as a generalisation of many other probabilistic models; e.g. dynamic Bayesian networks can model both hidden Markov models and linear dynamical systems (Murphy, 2002). This is extremely useful, as advances in one field can be passed to others, with the single formalism breaking down barriers of understanding.

The intuitive structural layout of a Bayesian network is also an asset when it comes to answering questions about the information stored in it. Taking advantage of this layout, algorithms have been developed that can answer probabilistic questions about variables in the face of uncertain evidence. This strictly numerical answer can easily be turned into a discriminative one as is given by many standard machine learning systems such as classifiers and clustering algorithms.

Aside from the theoretical aspects, the pragmatic use of Bayesian networks cannot be denied. They can be specified by experts, learnt from data, or be created from a combined approach. Much research has taken place in recent years. This, coupled with increased computational power, has meant that Bayesian networks can now be used in situations where they might have been seen as excessive. The list of applications is now too numerous to mention, but what was originally used in the domain of medical diagnosis and monitoring has spread to practically every area involving uncertainty.

1.1.2 Using Machine Learning Techniques to Construct Bayesian Networks

Along with the increased expressiveness of Bayesian networks comes a greater difficulty in specifying them. Bayesian networks can be built by hand, but this process is often seen as long and unwieldy (Abramson et al., 1996). Since Bayesian networks consist of two parts – the structure and the parameters – each of these must be specified separately.

Of the two, the structure is seen as the easier part for a human to give. The standard technique starts by getting an expert to specify the variables that are needed to model a given domain. This in itself is difficult, as an expert does not always know all the variables

that influence the concept. The experts might group the effects of different variables into one, or give variables that have a negligible effect on others. After the variables have been recognised, the arcs among the variables are normally specified by looking at causal relations among the variables – if X causes Y , then an arc is added from X to Y in the structure. Here, problems determining which variables are causes of other variables and problems with too many parents can crop up. These can be aggravated by missing or extra variables.

Once the structure has been given, the parameters must be specified. This task is normally hard for domain experts to do, as it is not natural for them to think in terms of conditional probabilities.

Machine learning techniques can help in this situation, as they are able to learn Bayesian networks from data. Indeed, much progress has been made in this field, with recent algorithms being able to provide good performance in reconstructing gold-standard Bayesian networks. Similar to specifying a Bayesian network by hand, the structure and parameters must be learnt separately. Unlike the hand-crafting procedures, learning the parameters of the network is by far the easier task. With complete data, only a single pass over the data is needed.

Learning the structure is more complicated. Since the space of possible structures is super-exponential in the number of variables, some form of heuristic must be used to find a good match to the data. However, most of these methods tend to be greedy in nature and therefore suffer from the well-known problem of local maxima in the learning space. In order to avoid these maxima, techniques which utilise randomness can be useful. One of these – ant colony optimisation and how it can be applied to this problem – is the subject of the next section.

1.1.3 Using Ant Colony Optimisation to Learn Bayesian Networks

A recent technique that can help with the problem of local maxima is that of ant colony optimisation (ACO) (Dorigo and Stützle, 2004). This metaheuristic is based on the swarming behaviour of ants searching for food and has shown good performance in solving many combinatorially hard problems. ACO has been successfully applied to such tasks as: the sequential ordering problem (Gambardella and Dorigo, 2000), the vehicle routing problem (Bullnheimer et al., 1999), the bin-packing problem (Levine and Ducatelle, 2004), classification rule induction (Parpinelli et al., 2002b) and many more

(Maniezzo and Colorni, 1999; Gambardella and Dorigo, 2000; Stützle, 1998; Costa and Hertz, 1997).

In ACO, a set of autonomous agents known as ants perform a walk of a construction graph, incrementally building a solution to the problem being looked at, e.g. a travelling salesman problem. The agents do not communicate directly; instead they rely on the principle of stigmergy, i.e. communicating via their environment. They do this by leaving a pheromone trail on the graph. This pheromone trail positively influences other ants to proceed in the direction that previously successful ants took. Inherent in this process is the idea of randomness; ants are free to decide which direction to go in, even if it seems that a particular direction is not optimal. In the end, sometimes taking the seemingly worse route is needed to get a better solution. This stigmergy principle is often coupled with a problem dependent heuristic. The heuristic induces ants towards better solutions and it is the combination of the pheromone and heuristic that produces good solutions.

When applied to the task of learning Bayesian network structures, ACO proceeds by having each ant move from graph structure to graph structure in the space of all possible structures. Because of the biased random nature of the metaheuristic, it is possible to bypass local maxima in the learning space and hence generate better solutions.

1.2 Thesis Objectives

The aim of this thesis is to investigate the effect of an implementation of the ACO metaheuristic to the problem of learning Bayesian network structures. To further this task, the following sub-aims were identified:

- To investigate current strategies in learning Bayesian network structures and identify problems in these strategies;
- To seek solutions to help in resolving these problems;
- To implement these solutions, in order that their effectiveness can be analysed;
- To set up a testing framework to see the performance of the solutions, both in regards to various parameter settings and against other strategies found in the literature;
- To analyse the results found and discuss any conclusions in the context of other strategies; and

- To test the implementations on a real-life problem involving input from domain experts.

1.3 Methodology

The main problem to be solved is that of learning the structure of a Bayesian network from data. In general, there are three types of methods that can achieve this. The first performs statistical tests on the data to identify conditional independencies and uses these independencies to build the structure (Spirtes et al., 2000).

The second uses what is known as a score-and-search approach. With this type of method, a search space is set up, with each of the states of the space being a Bayesian network structure. A scoring function is then specified, that can judge how good each state is. Because of the massive size of the space, an exhaustive search is not feasible, and heuristics must be used. Therefore, a set of operators is given that can modify a state to change it into a different state. With these elements, a search can proceed, by specifying a start state and repeatedly invoking operators on the current state to traverse the search space (Heckerman, 1995b).

The third method is similar to the second, in that a scoring function is needed to see how good a fit a structure is with the data. However, in this method, a search does not happen. Rather, dynamic programming is used to calculate the scores of all possible structures. The structure with the best score is then selected by the algorithm (Ott and Miyano, 2003).

1.3.1 Searching the Space of Equivalence Classes Using a Novel ACO Algorithm

The method developed in this thesis to solve the main problem as given above, is based on the second aforementioned type of search. This method is implemented in a new algorithm called ACO-E. A search space is set up, with each state of the space being an equivalence class of Bayesian network structures. This particular space avoids some of the problems associated with the space of structures, including a plateauing effect of the scoring function on statistically indistinguishable structures. A standard scoring function and set of operators are defined, so that a search can move through the space. The algorithm uses ant colony optimisation to find a good structure given the data provided. It does this by having each ant select an operator to apply to the current state. The operator

selected depends on a heuristic given by the scoring function and a pheromone value left by previous ants.

The ACO-E algorithm combines two ideas that have shown success in learning Bayesian network structures – that of searching through the space of equivalence classes (Chickering, 2002a) and that of using ant colony optimisation (de Campos et al., 2002a). By bringing these two designs together, it is hoped that a more accurate algorithm for learning structure can be produced.

1.3.2 Speeding up Search in the Space of Equivalence Classes

In developing the ACO-E algorithm, it was noticed that large amounts of time were spent in calculating a value needed to determine where a particular operator was valid. In order to reduce this time and hence speed up a search, methods were devised that cut down on the amount of checks that needed to be performed at each iteration of a search algorithm. Although these methods were developed for the purpose of speeding up the ACO-E algorithm, they are applicable to any search in the space of equivalence classes. This is especially true in the case of randomised algorithms that restart multiple times (e.g. a greedy search with multiple random starts) – of which ACO-E is one.

1.3.3 Testing the Hypothesised Methods

In testing the utility of the proposed methods, two main objectives were sought:

- Test the utility of the methods in themselves, e.g. how does varying the parameters of an algorithm change the output; and
- Test the utility of the methods compared to other methods, e.g. how does one algorithm compare to another in recreating gold-standard Bayesian networks that are used for testing purposes.

1.4 Preview of Results

In performing experiments involving the methods described above, results were found that supported their utility.

Firstly, the methods that were designed to speed up searching in the space of equivalence classes of Bayesian network structures were shown to have a significant effect. It was found that:

- the speed-up was by a factor of n , where n is the number of variables in the structure;
- this speed-up matched an analytical analysis of the computational complexity of the methods; and
- the effect is uniform across an algorithm run.

Secondly, the method used to improve the accuracy of learning a Bayesian network structure – namely the ACO-E algorithm – was found to significantly improve on the results of other algorithms. It was shown that:

- varying the parameters of ACO-E produced a significant effect on the output of the algorithm;
- ACO-E performed well against other algorithms that were similar in operation, e.g. that used ACO (but not in the space of equivalence classes) (de Campos et al., 2002a) or that searched in the space of equivalence classes (but did not use ACO) (Muruzábal and Cotta, 2004; Cotta and Muruzábal, 2004; Chickering, 2002a); and
- ACO-E performed well against other state-of-the-art Bayesian network structure learning algorithms, including MMHC (Tsamardinos et al., 2006), the sparse candidate algorithm (Friedman et al., 1999c), PC (Spirtes et al., 2000) and GES (Chickering, 2002b).

Finally, the experiments on the real-life data showed that ACO-E was able to reconstruct a regulatory network involving genes from the *Arabidopsis Thaliana* plant. It was discovered that:

- the false positive rate of the constructed networks was quite good;
- the scoring function used in the algorithm is often quite sensitive to one of its parameters;
- at small sample sizes, prior knowledge is necessary; and
- the algorithm was able to construct Bayesian networks with a good true positive rate.

1.5 Dissertation Structure

In following the aims laid out in Section 1.2, the structure of this document is as follows:

- Chapter 2 involves a literature review on Bayesian networks. This includes the foundations of the subject, the role played by inference of Bayesian networks, learning the parameters of Bayesian networks and learning the structure of Bayesian networks. In addition, a section on applications of Bayesian networks across many domains and a final comparison of various learning techniques is given.
- Chapter 3 is on ant colony optimisation. The basis of the technique is given along with a sample application, a look at ACO in the context of machine learning and two methods which use ACO in learning the structure of Bayesian networks.
- Chapter 4 contains the main new ideas of the thesis. These include:
 - an in-depth look at learning Bayesian network structures, by searching through the space of equivalence classes;
 - the methods used to speed up searching through this space; and
 - the implementation of the ACO-E algorithm, as mentioned above.
- Chapter 5 describes the experimental methodology used to examine the behaviour of the techniques described in Chapter 4. Descriptions of standard networks used in testing are given, as are the design of the experiments used to fulfil the thesis objectives, along with criteria with which to compare the results.
- Chapter 6 presents the results from the experiments run according to the methodology in Chapter 5. These results are discussed and tested using statistical techniques. These tests are used to discover whether the methods described in Chapter 4 improved on other methods described in the literature.
- Chapter 7 shows an experiment to test the utility of the ACO-E algorithm on a real-life problem. Data from experiments involving the *Arabidopsis Thaliana* plant were used to try and model its circadian clock. A simple extension of the ACO-E algorithm is proposed to model temporal data. Experiments used to check this utility are described, along with the results of the experiments and an evaluation of these results.

- Chapter 8 presents conclusions reached in performing the work described above. Along with these conclusions are examples of where this research could lead onto and possible directions to achieve these goals.

Chapter 2

A Review of the Literature on Learning Bayesian Networks

BAYESIAN NETWORKS have become a widely used method in the modelling of uncertain knowledge. Because of the difficulty domain experts have in specifying them, techniques that learn Bayesian networks from data have become indispensable. Recently however, there have been many important new developments in this field. This chapter, based on the work of Daly et al. (2008), takes a broad look at the literature on learning Bayesian networks – in particular their structure – from data. Specific topics are not focused on in detail, but it is hoped that all the major fields in the area are covered. An effort has been made to locate all the relevant publications, so that this thesis can be used as a ready reference to find the works on particular sub-topics. This chapter is not intended to be a tutorial – for this, there are many books on the topic, the most relevant probably being by Neapolitan (2004).

The chapter proceeds as follows. Firstly, the theory and definitions behind Bayesian networks are explained, in order that readers are familiar with the myriad terms that appear on the subject. Next, a brief overview of inference in Bayesian networks is presented. While this is not the focus of this thesis, inference is often used whilst learning Bayesian networks and therefore it is important to know the various strategies for dealing with the area. Thirdly, the task of learning the parameters of Bayesian networks – normally a subroutine in structure learning – is briefly explored. Fourthly, the main section on learning Bayesian network structures is given. Finally, a brief look at some applications of structure learning are examined and a comparison between different structure learning techniques is given.

2.1 Introduction to Bayesian Networks

Before properly beginning, it is useful to note that Bayesian networks are often known by other names. These include: recursive graphical models (Lauritzen, 1995), Bayesian belief networks (Cheng et al., 1997), belief networks (Darwiche, 2002), causal probabilistic networks (Jensen et al., 1990b), causal networks (Heckerman, 2007), influence diagrams (Shachter, 1986a) and doubtless many more. Compounding this confusion, authors often mean slightly different things when they use these terms. Nevertheless, the term Bayesian network seems to have become the prevalent way of describing this particular structure and it is how they will be described in this thesis.

Bayesian networks can have many different interpretations. This section hopes to capture their mathematical background. From this, the relations between Bayesian networks and other approaches to knowledge modelling can be seen. To start out with, a very short introduction will be given on probability theory, Bayes' rule and conditional independence. These ideas are fundamental to the theory of Bayesian networks, and will enable a better understanding of the context of the subject.

2.1.1 Preliminaries

Most people have an intuitive understanding of probability as either the long run limit of a series of random experiments, or a subjective belief of what is likely to happen in a given situation. To introduce this topic in a more rigorous manner, a short background will be given here in order to introduce terminology and notation. To start with, a *sample space* Ω is defined as a set of *outcomes*, i.e. $\Omega = \{\omega_1, \omega_2, \dots, \omega_n\}$. An *event* E on Ω is a subset of Ω , i.e. $E \subseteq \Omega$. From this point of view, outcomes may be seen as elementary events, i.e. events that can only take on a true/false character. Events are things which we might be interested in and tend to be the fundamental unit of probability theory. A probability distribution P , is a function from the space of events to the space of real numbers from 0 to 1, i.e. $P: \mathbb{P}(\Omega) \rightarrow [0, 1]$, where $\mathbb{P}(\Omega)$ is the power set of Ω . So when we say the probability of an event E is 0.76, we are saying $P(E) = 0.76$. Since events are sets, we can perform set operations on them. This allows us to specify the probability of two events, E and F occurring, by $P(E \cap F)$. From this we can define another very useful idea, that of conditional probability.

The *conditional probability* of an event E occurring, given that an event F has occurred

is given by

$$P(E|F) = \frac{P(E \cap F)}{P(F)}$$

Obviously for this to be defined, $P(F)$ must be strictly positive. As an aside, it should be noted that

$$P(E \cap F) = P(E|F)P(F) = P(F|E)P(E)$$

This implies that

$$P(E|F) = \frac{P(F|E)P(E)}{P(F)}$$

This is the well-known *Bayes' formula* and is in itself fundamental to many modern statistical techniques in machine learning. The term $P(E|F)$ is often known as the *posterior probability* of E given F . The term $P(F|E)$ is often referred to as the *likelihood* of E given F , when viewed as a function of E and the term $P(E)$ is the *prior* or *marginal* probability of E . The term $P(F)$ is a normalising term that is often expanded out as

$$P(F) = \sum_{H_i \in H} P(F \cap H_i) = \sum_{H_i \in H} P(F|H_i)P(H_i),$$

where H is a set of pairwise disjoint events H_i such that $H_1 \cup H_2 \cup \dots \cup H_n = \Omega$. The reason for this expansion is that the terms $P(H_i|F)$ and $P(H_i)$ are often much easier to obtain than $P(F)$.

Given the definition of conditional probability, we can now define what it means for events to be independent. Two events, E and F are independent if

$$P(E|F) = P(E) \quad \text{and} \quad P(F|E) = P(F)$$

If $P(E)$ and $P(F)$ are both positive, then both equations imply the other. This definition leads to that of conditional independence, which will involve a third event. Two events, E and F are conditionally independent, given another event G if

$$P(E|F \cap G) = P(E|G) \quad \text{and} \quad P(F|E \cap G) = P(F|G)$$

Again, these are equivalent if $P(E)$, $P(F)$ and $P(G)$ are strictly positive. The notion of conditional independence is central to Bayesian networks and many other models dealing with probabilistic relationships. It is often given its own notation as $I_P(E, F|G)$ which means event E is conditionally independent of event F given event G , under probability distribution P .

To complete this subsection, it suffices to explain the concepts of random variables and joint probability. In common parlance, random variables are variables that can take on a

value from a given set, and have a certain probability associated with taking on this value. Technically, a *random variable* X is a function from a sample space Ω to a *measurable space* M . To illustrate how they are used in practice, imagine the following scenario. Say we are dealing with temperature and we have three different measures of it, *low*, *medium* and *high*. We could then state that the random variable X stands for temperature and our measurable space M is the set $\{low, medium, high\}$. So when we make the statement $P(X = low)$, the probability that the temperature is low, the expression $X = low$ is an event E . Therefore, we are calculating $P(E)$, such that $E = \{\omega | \omega \in \Omega, X(\omega) = low\}$. Normally, we leave all the details of sample space and probability measure implicit and never mention them. Instead we deal directly with random variables, but it is beneficial to know where the notation comes from.

Finally, the *joint distribution* of a set of random variables is the multidimensional analogue of the single variable case. For example, $P(X, Y)$ is the joint distribution of two random variables X and Y . To specify a probability for an event, we assign values to the variables. $P(X = x, Y = y)$ is the probability that X takes on value x and Y takes on value y . We can *marginalise* across some of the variables by adding up across all possible values of those variables. For example, given $P(X, Y)$ we can get the probability distribution $P(X)$ by

$$P(X) = \sum_{y \in M(Y)} P(X, Y = y)$$

where $M(Y)$ is the domain (or measure space) of Y . It is useful to note that with the notation $P(x, y)$, where x and y are lower case letters, there are usually implied random variables, so that $X = x$ and $Y = y$, i.e. $P(x, y) \equiv P(X = x, Y = y)$.

2.1.2 Bayesian Networks

To see why conditional independence is important, imagine the following scenario. Imagine we wanted to define a joint probability distribution across many variables $P(X_1, X_2, \dots, X_n)$. If each variable is binary valued, then we need to store $2^n - 1$ values. It should be obvious that with this storage requirement exponential in the number of variables, things soon become intractable. To get around this, firstly note the identity

$$P(X_1, X_2, \dots, X_n) = P(X_1 | X_2, X_3, \dots, X_n) P(X_2, \dots, X_n).$$

Now, say that $I_P(X_1, \{X_3, \dots, X_n\} | X_2)$, i.e. X_1 is independent of the rest of the variables given X_2 . Then,

$$P(X_1, X_2, \dots, X_n) = P(X_1 | X_2) P(X_2, \dots, X_n).$$

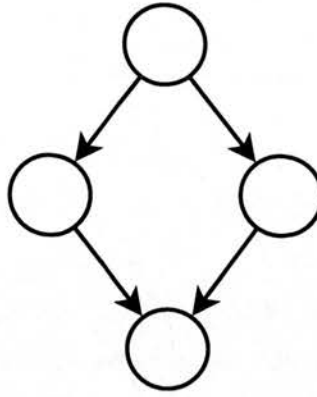


Figure 2.1: A directed acyclic graph

Notice how the expression involving X_1 has become much shorter and we have a slightly smaller joint term (minus X_1). If we can find conditional independencies for the rest of the variables such that this factorisation can proceed in a chain like fashion, we will be left with a product of terms, each of which will only contain (hopefully) a small number of random variables. Then, to specify the joint probability distribution, we need only specify a number of conditional probability distributions. The reason for this is two-fold. Firstly, if each variable is conditionally independent of most others, then we only need specify a small number of values for each distribution. Secondly, humans generally find it easier to specify the values of a conditional distribution.

There are many statistical models that take advantage of these properties. Examples can be found in the paper by Lauritzen and Wermuth (1989) and the books by Castillo et al. (1997a), Pearl (1988) and Whittaker (1990). The particular model that will be dealt with here is the Bayesian network. Before defining what they are, some definitions relating to graphs will be given.

A graph \mathcal{G} is given as a pair (V, E) , where $V = \{v_1, \dots, v_n\}$ is the set of vertices or nodes in the graph and E is the set of edges or arcs between the nodes in V . A directed graph is a graph where all the edges have an associated direction from one node to another. A directed acyclic graph or DAG, is a directed graph without any cycles, i.e. it is not possible to return to a node in the graph by following the direction of the arcs. For illustration, the graph in Figure 2.1 is a DAG. The parents of a node v_i , $Pa(v_i)$, are all the nodes v_j such that there is an arrow from v_j to v_i ($v_j \rightarrow v_i$). The descendants of v_i , $D(v_i)$, are all the nodes reachable from v_i by following the arrows repeatedly. The non-descendants of v_i , $ND(v_i)$, are all the nodes that are not descendants of v_i .

Let there be a graph $\mathcal{G} = (V, E)$ and a joint probability distribution P over the nodes

in V . Say also that the following is true

$$\forall v \in V I_P(\{v\}, ND(v) | Pa(v))$$

That is, each node is conditionally independent of its non-descendants, given its parents. Then it is said that \mathcal{G} satisfies the Markov condition with P , and that (\mathcal{G}, P) is a Bayesian network. Notice the conditional independencies implied by the Markov condition. They allow the joint distribution P to be written as the product of conditional distributions; $P(v_1, v_2, \dots, v_n) = P(v_1 | Pa(v_1)) P(v_2 | Pa(v_2)) \dots P(v_n | Pa(v_n))$. However, more importantly, the reverse can also be true. Given a DAG \mathcal{G} and either discrete conditional distributions or certain types of continuous conditional distributions (e.g. Gaussians), of the form $(P(v_i | Pa(v_i)))$ then there exists a joint probability distribution $P(v_1, v_2, \dots, v_n) = P(v_1 | Pa(v_1)) P(v_2 | Pa(v_2)) \dots P(v_n | Pa(v_n))$. This means that if we specify a DAG – known as the structure – and conditional probability distributions for each node given its parents – known as the parameters –, we have a Bayesian network, which is a representation of a joint probability distribution.

It may be wondered whether there are any other conditional independencies that may be obtained from the Markov condition. It turns out there are and these can be identified by a property known as *d-separation*, which is a purely graphical test, i.e. a test that can be implemented by performing a search on a graph. The notation $I_{\mathcal{G}}(A, B | C)$ means that the nodes in set A are d-separated from the nodes in set B , given set C . It is also the case that given the Markov condition, d-separation identifies conditional independencies *only* in P . That is, $I_{\mathcal{G}}(A, B | C) \implies I_P(A, B | C)$ for all mutually disjoint subsets A, B and C of V . If a graph \mathcal{G} can be found such that $I_{\mathcal{G}}(A, B | C) \iff I_P(A, B | C)$, then it is said that \mathcal{G} is faithful to P . This is important because it implies that the arcs in the graph directly model dependencies between variables, whereas up to now only independencies have been discussed. This brings the structure of the Bayesian network closer to human intuition, in that an arc between two nodes implies there is a direct relation between those variables.

Finally, if it is assumed that in a Bayesian network, an arc from x to y means that x is a direct cause of y , then one of a number of *causal assumptions* is being made (see Druzdzel and Simon (1993), Huang and Valtorta (2006) and Neapolitan (2004) for more on these assumptions). If this is the case, then this Bayesian network is capturing knowledge in a succinct way that is immediately obvious to humans, yet also with a well understood formalism underlying the operations that can be performed. It is for these reasons that Bayesian networks are so popular. As mentioned before, there exist other structures that

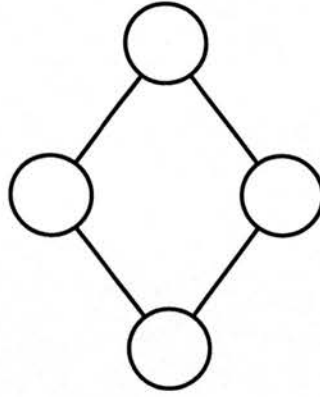


Figure 2.2: The skeleton of the DAG in Figure 2.1

model conditional independencies, such as Markov fields, that seem to be less popular because of their opaqueness. For a more in-depth look at the differences and similarities of these structures, see the paper of Smyth (1997). Also, for a look at the explanatory properties of Bayesian networks see the papers of Druzdzel (1996) and Madigan et al. (1997). The relationship between Bayesian networks and causality is sometimes fraught, but there are methods as described in Section 2.4.7, that mean a causal interpretation is valid. For more on the intersection of Bayesian networks and causal models see the books of Glymour and Cooper (1999), Spirtes et al. (2000) and Pearl (2000).

2.1.3 Markov Equivalent Structures

For the purposes of this thesis, it is necessary to define some further terms relating to the structures of Bayesian networks. These terms arise because of redundancies in the DAG representation of the structure.

It has been known for some time, that there are DAGs that are equivalent to one another, in the sense that they entail the same set of conditional independencies as each other, even though the structures are different. According to a theorem by Verma and Pearl (1991), two DAGs are equivalent iff they have the same skeletons and the same v-structures. By skeleton, it is meant the undirected graph that results from undirecting all edges in a DAG and by v-structure (sometimes referred to as a morality), it is meant a head-to-head meeting of two arcs, where the tails of the arcs are not joined. These concepts are illustrated in Figures 2.2 and 2.3. From this notion of equivalence, a class of DAGs that are equivalent to each other can be defined, notated here as $Class(\mathcal{G})$.

To represent the members of this equivalence class, a different type of structure is used, known as a partially directed acyclic graph (PDAG). A PDAG (an example of which

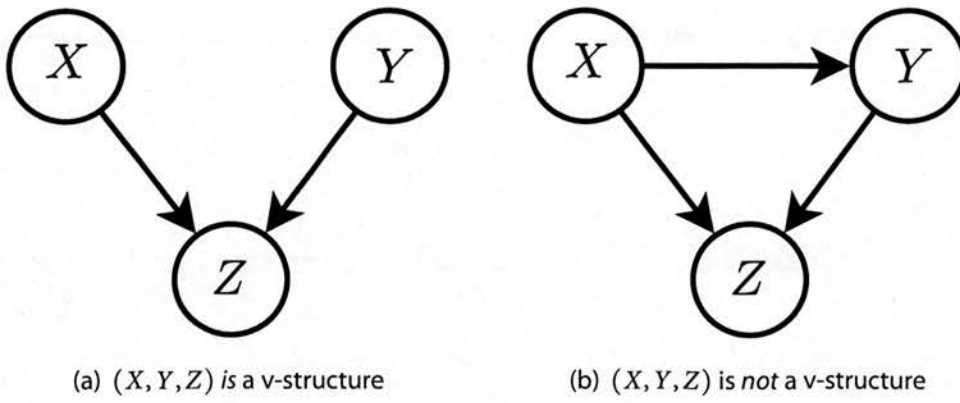


Figure 2.3: V-Structures

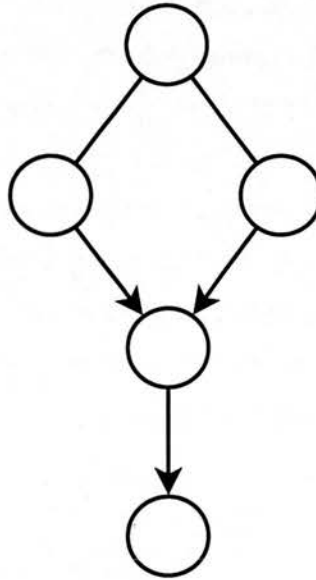


Figure 2.4: A partially directed acyclic graph

is shown in Figure 2.4) is a graph that contains both undirected and directed edges and that contains no directed cycles and will be notated herein as \mathcal{P} . The equivalence class of DAGs corresponding to a PDAG is denoted as $Class(\mathcal{P})$, with a DAG $\mathcal{G} \in Class(\mathcal{P})$ iff \mathcal{G} and \mathcal{P} have the same skeleton and same set of v-structures.

Related to this is the idea of a *consistent extension*. If a DAG \mathcal{G} has the same skeleton and the same set of v-structures as a PDAG \mathcal{P} then it is said that \mathcal{G} is a consistent extension of \mathcal{P} . Not all PDAGs have a DAG that is a consistent extension of itself. If a consistent extension exists, then it is said that the PDAG *admits* a consistent extension. Only PDAGs that admit a consistent extension can be used to represent an equivalence class of DAGs and hence a Bayesian network. An example of a PDAG that does not have a consistent

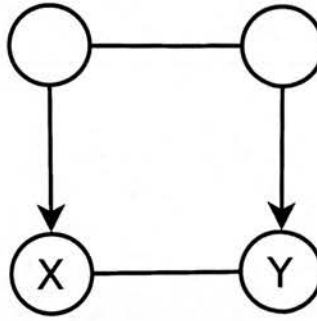


Figure 2.5: A PDAG for which there exists no consistent extension

extension is shown in Figure 2.5. In this figure, directing the edge $x - y$ either way will create a v-structure that does not exist in the PDAG and hence no consistent extension can exist.

Directed edges in a PDAG can be either:

- compelled, or made to be directed that way; or
- reversible, in that they could be undirected and the PDAG would still represent the same equivalence class.

From this idea, a completed PDAG (CPDAG) can be defined, where every undirected edge is reversible in the equivalence class and every directed edge is compelled in the equivalence class. Such a CPDAG will be denoted as \mathcal{P}^c . It can be shown that there is a one-to-one mapping between a CPDAG \mathcal{P}^c and $Class(\mathcal{P}^c)$. Therefore, by supplying a CPDAG, one can uniquely denote a set of conditional independencies. This can be useful in defining certain strategies to learn Bayesian network structures from sets of data, as seen in Section 2.4.5. For a more in-depth look at this topic, see the papers of Andersson et al. (1997) and Chickering (1995).

Equivalence classes of Bayesian network structures are particularly relevant for this thesis, as they are the basis of the search space on which the search will proceed. This will be explained further in Section 4.1.1.

2.1.4 Special Types of Bayesian Networks

There exist certain specialisations of Bayesian networks that deal with situations that demand slightly more structure than the general Bayesian network. A brief summary of these types will be given here.

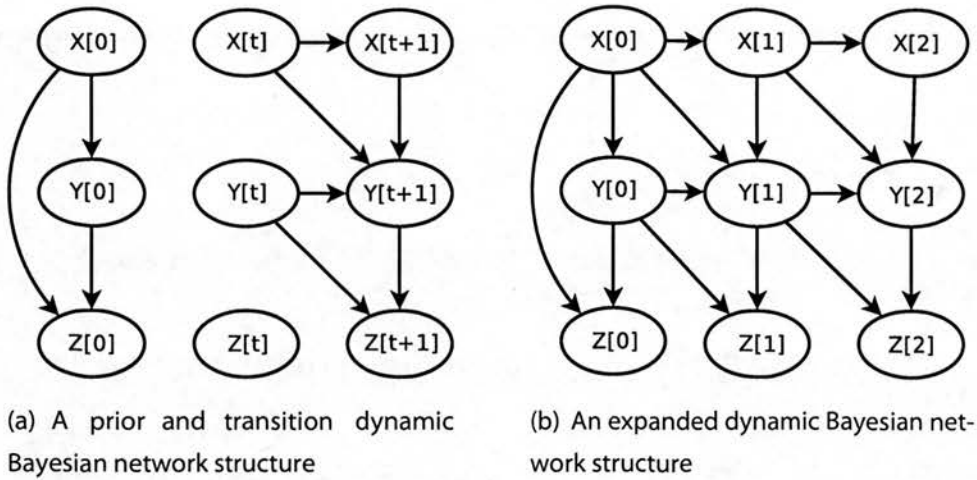


Figure 2.6: Dynamic Bayesian network structures

2.1.4.1 Causal Interaction Models

Otherwise known as causal independence models, they imply that the parents of nodes in a Bayesian network are independent of each other, to some degree. Coming in various flavours, the best-known type is the noisy-OR model as defined by Kim and Pearl (1983) and showcased in Pearl (1988). This was later generalised by Srinivas (1993) to multiple causes and arbitrary combination functions. Heckerman and Breese (1996) and Meek and Heckerman (1997) also have a look at the field in the context of inference and learning.

2.1.4.2 Dynamic Bayesian Networks

In order to model temporal processes, special structures are needed. This is because the arcs in a Bayesian network say nothing about time, only about probabilistic relationships. For these purposes, dynamic Bayesian networks (DBNs) are a useful representation. The key to DBNs is that they are specified in two parts, a prior Bayesian network that specifies the initial conditions and a transition Bayesian network that specifies how variables change from time to time. An example DBN, due to Friedman et al. (1998), is shown in Figure 2.6(a). In this, the prior and transition network are shown. It can be seen that whilst the prior network is simply a general Bayesian network, the transition network has slightly more structure to it. In this, there are two layers of nodes, and arcs from the first layer only go to the second. Also, no arcs go from the second layer to the first. For the purposes of performing inference, or simply reasoning about them, DBNs can be expanded out into a single network. The network in Figure 2.6(a) has been expanded out in Figure 2.6(b). More information of DBNs can be found in the papers of Friedman

et al. (1998) and Dean and Kanazawa (1989) and in the work of Murphy and Mian (1999). Also, Ghahramani (1998) examines the topic from the perspective of learning.

In this thesis, dynamic Bayesian networks are used in Chapter 7 in order to model the temporal behaviour of the circadian clock of the plant *Arabidopsis Thaliana*.

2.1.4.3 Influence Diagrams

By themselves, Bayesian networks do not specify what to do in a particular situation; they only say what is the probability of certain things happening. If a Bayesian network is augmented with two other types of nodes, then it is possible for actions to be decided based on given evidence. These two types of nodes are utility nodes and decision nodes. Utility nodes represent the value of a particular event, whilst decision nodes represent the choices that might be made.

Influence diagrams represent a powerful formalism in helping to make decisions under uncertainty. They can be used in static situations such as diagnosis or dynamic situations when combined with DBNs, such as controllers. More information can be found in the articles of Shachter (1986a, 1988).

2.2 Inference in Bayesian Networks

Whilst performing inference in Bayesian networks is a large topic in its own right, any treatment of Bayesian network structure learning has to have at least some mention of the subject. This is because inference is often a subroutine in structure learning problems, especially in the case of missing data or hidden nodes. Therefore, a short summary will be given of the major methods of performing inference, in order that a full appreciation can be had of this expansive area.

The summary will contain a short introduction on what inference is, followed by a look at various techniques used to solve the problem. This starts with the message passing algorithm of Pearl (Section 2.2.2), probably the most important base technique and moves on to deal with the problems created by multiply connected networks (Section 2.2.3). The exact techniques covered include clustering (Section 2.2.4), conditioning (Section 2.2.6), node elimination and arc reversal (Section 2.2.5), symbolic probabilistic inference (Section 2.2.7) and polynomial compilation (Section 2.2.8). The various approximate methods include Monte-Carlo methods (Section 2.2.9), search based approximation (Section 2.2.10.1), model simplification (Section 2.2.10.2) and loopy belief propagation (Section 2.2.10.3). Finally, special topics such as inference in dynamic Bayesian networks,

causal independence networks and robustness of inference will be looked at. For a good survey of the literature, see the paper by Guo and Hsu (2002).

There are many books that deal with Bayesian network inference. Some of the more popular ones are the original by Pearl (1988), the knowledge-focused book by Castillo et al. (1997a) and the up-to-date book by Jensen and Nielsen (2007). Other books include those by Cowell et al. (1999), Korb and Nicholson (2004) and Neapolitan (2004).

2.2.1 Introduction to Inference

Inference in Bayesian networks generally refers to:

- finding the probability of a variable being in a certain state, given that other variables are set to certain values; or
- finding the set of variables that best explains why a set of other variables are set to certain values.

The Bayesian network structure in Figure 2.7 will be used to illustrate these problems. This is the well-known ASIA network, as defined by Lauritzen and Spiegelhalter (1988). With the first problem, a patient might present as a smoker and obtain a positive X-ray. Using this network, a physician might want to find out the probability that they have lung cancer i.e. $P(\text{lung cancer} = \text{true})$. With the second problem, a physician might want to find out the most probable explanation that explains these symptoms, i.e. what is most likely to have caused the symptoms. In this thesis, it is generally the former problem that is being looked at, though the latter will be mentioned as well.

2.2.2 Trees and Polytrees

The first Bayesian network inference algorithms were developed for trees and polytrees, i.e. Bayesian network structures that contained only a single path between any two nodes. Pearl (1982) was the first to apply an inference procedure on trees, with Kim and Pearl (1983) extending this to polytrees (i.e. graphs with no loops). The polytree algorithm was later extended by Peot and Shachter (1991) to visit each node at most twice. Regardless of any speed-ups, Pearl's message passing algorithm is important, as it operates in polynomial time with singly connected networks. An illustration of this scheme is shown in Figure 2.8. Here, each node is an autonomous processor that collects evidence from its n parents $(\pi_x(u_1), \dots, \pi_x(u_n))$ and m children $(\lambda_1(x), \dots, \lambda_m(x))$,

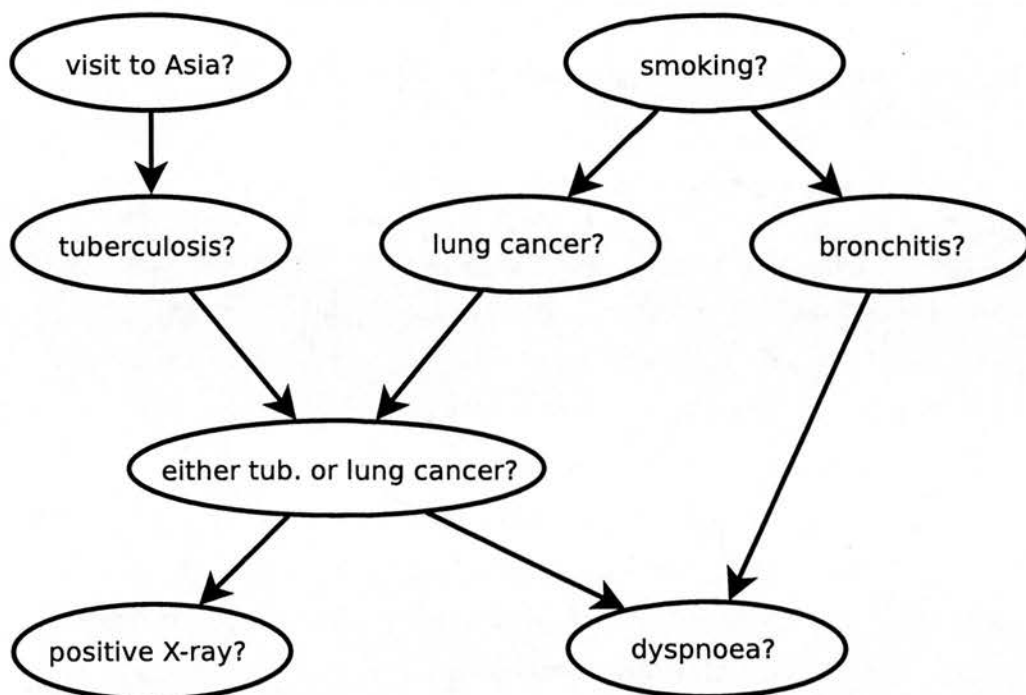


Figure 2.7: The ASIA Bayesian network structure

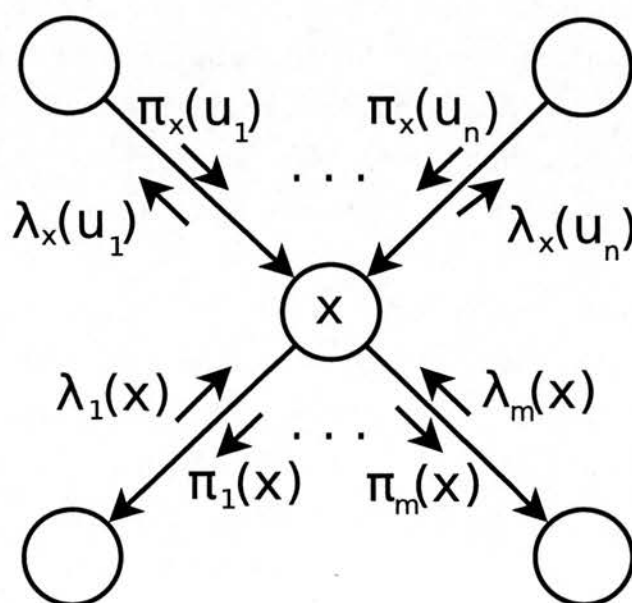


Figure 2.8: Inference by message passing

performs processing and sends out messages to its parents ($\lambda_x(u_1), \dots, \lambda_x(u_n)$) and children ($\pi_1(x), \dots, \pi_m(x)$). The whole procedure is inherently asynchronous and is the basis of many of the inference schemes for multiply connected networks.

2.2.3 Multiply Connected Networks

A problem with Pearl's algorithm is that it can only be applied to singly connected networks. Otherwise its messages can loop forever. Pearl (1986b) reported on this problem and mentioned some techniques that can solve this, which are explained in the next sections. Because of the large number of possible techniques, the comparison of Díez and Mira (1994) is quite helpful.

The probable explanation for the plethora of inference methods is that Bayesian network inference is NP-hard in both the exact (Cooper, 1990) and approximate (Dagum and Luby, 1993) case, where the network is multiply connected. The following techniques seek to cut down the possibly exponential time needed.

2.2.4 Clustering

One of the first methods to help apply the message passing algorithm to multiply-connected networks was by Spiegelhalter (1986). In this he describes a way of 'pulling loops together', into clusters. These clusters are then joined together into a singly connected structure, and a message-passing algorithm is started. This is built upon by Lauritzen and Spiegelhalter (1988) and then by Jensen et al. (1990a), who describe a variant of the clustering algorithm that builds a so called *junction tree*. They later give an optimal algorithm for junction tree construction given a triangulated graph (Jensen and Jensen, 1994).

Later authors looked into trying to optimise junction tree inference. Breese and Horvitz (1991) show how to trade off time spent on decomposition of the Bayesian network against actual inference. Other authors look at ways to get an optimal decomposition, e.g. Kjærulff (1992b) uses simulated annealing, Gámez and Puerta (2002) use ant colony optimisation in building the tree and Huang and Darwiche (1996) show how best to implement clustering. Some useful bounds have been found by Becker and Geiger (2001, 1996b), who give an algorithm that is sufficiently fast for building close to optimal junction trees.

Other authors have looked at the structure of the clique tree; Kjærulff (1997) shows how the cliques in the tree may themselves be factored into a clique tree, and Darwiche

(1998) shows how to keep clique trees up to date after pruning irrelevant parts of the network.

A clustering architecture that differs slightly from Lauritzen and Spiegelhalter and Jensen et al. is that of Shenoy and Shafer (1990) and Shafer and Shenoy (1990). It is worth knowing about, as it has been used by various authors, albeit to a lesser degree than the other schemes e.g. in (Shenoy, 1997) and (Schmidt and Shenoy, 1998).

2.2.5 Variable Elimination and Arc Reversal

A simple method of inference involves reversing arcs in a Bayesian network and removing variables. Shachter (1986a,b) introduced this in the context of evaluating influence diagrams – Bayesian networks that have decision and utility nodes that recommend a course of action to follow. This idea is continued on in (Shachter, 1988). It is useful to note that the node removal method of Zhang and Poole (1994b) proceeds from a different angle than Shachter.

2.2.6 Conditioning

Another one of the original techniques used to perform inference in multiply connected networks was that of conditioning. In this procedure, loops in the network are broken by instantiating nodes and the message passing algorithm is run on the singly connected networks, one for each combination of values that the nodes take on. Pearl (1986a) was the first to use this method, whilst Suermondt and Cooper (1988, 1990) show the optimal cutset is NP-hard to find. One issue with conditioning is that the set of nodes that cut the loops (the cutset) need to have a joint prior probability assigned to them; Suermondt and Cooper (1991) have a method to handle this.

Because conditioning is NP-hard it can be good to know that Becker and Geiger (1996a, 1994) have an algorithm (MGA) that finds a loop cutset with a guaranteed cardinality of less than twice the minimum cardinality. Other researchers have designed methods to try to alleviate the problems of conditioning; for more information see e.g. (Díez, 1996), (Shachter et al., 1994), (Darwiche, 1995) and (Darwiche, 2001b).

2.2.7 Symbolic Probabilistic Inference

Li and D'Ambrosio (1994) have found a method that splits the task of inference into two parts. Firstly, a symbolic factorisation of the joint probability distribution based

on the Bayesian network is found. Then a numeric step is performed where the actual probabilities are calculated. This style of inference has been built on by Chang and Fung (1995), who look at continuous variables and by Castillo et al. (1996, 1995) who develop a slightly different system for the symbolic inference.

2.2.8 Polynomial Compilation

A recent technique by Darwiche (2003) and Park and Darwiche (2004) shows that Bayesian networks can be represented as a polynomial. Probabilistic queries can be formulated by evaluating and differentiating this polynomial. This is based on the fact that every Bayesian network is a multi-linear function, which can be encoded as a decomposable negation normal form (d-DNNF) (Darwiche, 2001a), a language for representing propositional statements that has useful properties for evaluation. This can then be implemented as an arithmetic circuit (Darwiche, 2002), which is easy to evaluate and differentiate.

The inference of Bayesian networks as polynomials is interesting, as it can be shown that they subsume other methods of inference such as clustering. They can also be more efficient than other methods that have been discussed, such as clustering and conditioning, as the compilation phase of the method can be performed offline and optimisations performed (Chavira and Darwiche, 2007).

2.2.9 Monte-Carlo Methods

Because inference in Bayesian networks was found to be NP-hard in general (Cooper, 1990), attention was paid to heuristic and stochastic techniques to help solve the problem. It was then found that approximate inference is also NP-hard (Dagum and Luby, 1993). However, in general, approximate inference techniques have a wider range of applicability on hard networks than exact techniques. Some of the most prevalent inexact techniques are based on Monte-Carlo methods; the paper of Cousins et al. (1993) has a short tutorial on the subject in relation to Bayesian network inference, whilst the paper of Dagum and Horvitz (1993) analyses the performance of simulation algorithms using a Bayesian perspective.

2.2.9.1 Logic Sampling

One of the first techniques to use Monte-Carlo methods was introduced by Henrion (1988). In this, nodes are instantiated in topological order. The particular instantiation

depends on the probability distribution of that node. If, on an evidence node, the instantiation does not match, then that instantiation is discarded. When this procedure is iterated, each node will have been instantiated to each of its values a certain number of times and from this the probability can be estimated. However, there is a problem with this, in that if the evidence is unlikely, a large number of samples may be discarded. This can mean it takes a long time to get a reasonable estimate.

Various authors have suggested ways to mitigate the problem of unlikely evidence. The first of these were by Fung and Chang (1990) and Shachter and Peot (1990) who discussed a strategy called likelihood weighting, that does not discard evidence. This strategy was examined by Shwe and Cooper (1991) on a dense medical Bayesian network. Likelihood weighting is a very simple strategy and because of this, can often outperform more complicated strategies such as Gibbs sampling and other approximation schemes as discussed below.

From this point on, authors examined ways to improve this type of sampling approach. Examples include those of Bouckaert (1994c); Bouckaert et al. (1996) and Cano et al. (1996) who look at ways to more evenly sample the space. Following on from this, systems have been demonstrated by Pradhan and Dagum (1996), Dagum and Luby (1997) and Hernández et al. (1998). Some of the newest work is by Cheng and Druzdzel (2000, 2001) with their AIS-BN system, which has good performance characteristics across a wide range of classes, guaranteed bounds on inferred probabilities and a simple stopping rule.

2.2.9.2 Markov Chain Monte-Carlo Methods

As well as straight forward logic sampling schemes, authors have looked to other methods such as Gibbs sampling. Examples include early schemes such as that of Pearl (1987) and that of Chavez and Cooper (1990), whose algorithm has computable bounds. However, the complexity of these methods, compared to the likelihood weighting inspired approaches, means they are rarely used in practice.

2.2.10 Other Approximate Inference

As well as sampling based approaches, inference in Bayesian networks may be tackled using other, more heuristic methods. These include search based methods, model simplification methods and ones based on the loopy belief propagation idea, which will be explained later. A comparison of sampling and search based algorithms in approximate inference can be found in (Lin and Druzdzel, 1999).

2.2.10.1 Search Based Approximation

Search based approximations look for a small fraction of high probability instantiations and use them to approximate the distribution. Like sampling methods they have the advantage of being anytime, but can also keep the approximation in the form of guaranteed bounds, which might be important in certain contexts such as real-time systems.

An early example of these is by Poole (1993a) who demonstrates an algorithm that computes the exact answer if run to completion, but can be stopped to obtain a bound. This is extended so that it works best in distributions that are highly skewed (Poole, 1993b, 1996). Another author who shows that search can work well with skewed distributions is Druzdzel (1994). For later work on this style of technique, see the works of Monti and Cooper (1996), Santos et al. (1996, 1997), Shimony and Santos (1996) and Santos and Shimony (1998).

2.2.10.2 Model Simplification

Another class of approximations works by simplifying the model being queried. E.g. Kjærulff (1994, 1993) shows how to remove edges from the moralised independence graph, whilst constructing a clique tree. Wellman and Liu (1994) propose reducing the number of states of a node to reduce computation time. Draper and Hanks (1994) compute interval bounds by examining a subset of the nodes. This can get more accurate as the subset increases. van Engelen (1997) simply removes arcs from the network and then uses exact techniques. Other authors describe removing nodes from the network (Poole, 1997, 1998; Poole and Zhang, 2003; Jaakkola and Jordan, 1997). Finally, authors have recently started to use variational methods to approximate the model and then use exact inference (Bishop et al., 1998; Jaakkola and Jordan, 1999a,b; Jordan et al., 1999).

2.2.10.3 Loopy Belief Propagation

The final form of approximate inference procedures that will be looked at is based on loopy belief propagation. This method involves message passing in the multiply connected graph. In some cases, it can work well, e.g. in the case of a single loop as shown by Weiss (2000). However in general, it does not always work well (Murphy et al., 1999). From this perspective, Yedidia et al. (2001) and Pakzad and Anantharam (2002) have created generalised versions that have better convergence when faced with loops.

2.2.11 Inference in Dynamic Bayesian Networks

Inference in dynamic Bayesian networks often needs a special approach to deal with their particular structure. Although a transition DBN can be represented as a finite number of time slices (normally two), inference in general needs to be computed over the expanded network; i.e. inference needs to be computed at a *particular* time. Apart from the possibly massive number of nodes if the time is far in the future, the repetitious structure of this expansion is often not amenable to standard exact techniques for multiply-connected networks; see Boyen and Koller (1998) for a look at this problem and a possible solution. Kjærulff (1992a) looks at reasoning in dynamic Bayesian networks, based on Lauritzen and Spiegelhalter's approach, whilst Ghahramani and Jordan (1997) use variational approximations on factorial hidden Markov models (a subtype of DBNs). Meanwhile Kanazawa et al. (1995) adapt standard sampling techniques to the 'special characteristics' of DBNs.

2.2.12 Causal Independence Networks

Bayesian networks are often specified, where all parents of a node are independent of each other. This can happen if the network was constructed by hand, or if in the course of structure learning, prior knowledge specified that this should be the case. Therefore, inference procedures need to be aware of this possible situation. An advantage is that causal independence models can reduce inference complexity (Zhang and Poole, 1994a).

Inference in causal independence networks has been performed since Kim and Pearl (1983) specified their extension of Pearl's message passing scheme. From then on, authors have developed different methods of representing causal independence and how to perform inference and learning. For example, Zhang and Poole (1996) examine methods involving an operator acting upon the effects of a node's parents, e.g. *or*, *sum* or *max*. Jaakkola and Jordan (1996) look at computing upper and lower bounds on likelihoods in sigmoid and noisy-OR networks. Huang and Henrion (1996) also investigate noisy-OR inference with their TopEpsilon system. Other interesting papers on the subject include those by Heckerman and Breese (1996), Boutilier et al. (1996) and Zhang and Yan (1998).

2.3 Learning Bayesian Network Parameters

Whilst learning the parameters in a Bayesian network is an important task in itself, it is also significant in the context of learning the structure of a Bayesian network. This

is because most structure learning – particularly that using a scoring paradigm, as illustrated in Section 2.4.4 – has parameter learning as a subroutine. That is not to say that in learning a structure, parameters need to be explicitly represented and learnt. It is that the fundamental ideas behind them are the same and they are, in a sense, manifestations of the same procedures.

The parameters that are learnt in a Bayesian network depend on the assumptions that are made about how the learning is to proceed. For example, in the case of maximum likelihood learning, the parameters could be the actual probabilities in the conditional probability table attached to each node. Whereas in a Bayesian setting, the parameters could be used to specify a conditional density that in turn models the probabilities in a conditional probability table.

Fitting parameters to a model has mostly been attacked from the point of view of statistical machine learning. Good background material on the matter can be found in Whittaker (1990), but a more directed look is given by Spiegelhalter and Lauritzen (1990). For a gentle and broad introduction, the book of Neapolitan (2004) and article of Buntine (1994) are quite readable, whilst parameter learning in the context of structure learning is seen in (Heckerman et al., 1995).

2.3.1 Multinomial Data

A multinomial variable is a variable that can take on one of a finite number of possible values. Any data corresponding to a multinomial variable is known as multinomial data. When dealing with multinomial data, there are choices that can be made as to how the learning is to proceed. Perhaps one of the simplest methods is to estimate the parameters of the model using a maximum likelihood approach. However, this has a problem with sparse data, in that some probabilities – perhaps most of them – can be undefined if a case does not come up in the database. This can cause problems later with inference. To counteract this, some form of prior distribution is normally placed on the variables, which is then updated from the data. An example of this would be a distribution that said all values of a particular variable were of an equal prior probability to begin with, but changed quickly to reflect the observed data. Heckerman and Geiger (1995) and Buntine (1996) discuss this more. Also, under certain reasonable assumptions – that the parameters of the network are independent of each other and the density function characterising each parameter is strictly positive – Geiger and Heckerman (1995, 1997) showed that this distribution must be Dirichlet. The Dirichlet distribution is

the multivalued generalisation of the Beta distribution and is a conjugate prior of the multinomial; i.e. when updated with new information, the updated distribution is again Dirichlet. As an example, the form of the Beta density function is given by

$$f(x; \alpha, \beta) = \frac{x^{\alpha-1} (1-x)^{\beta-1}}{\int_0^1 u^{\alpha-1} (1-u)^{\beta-1} du},$$

for parameters α and β and variable x . When a series of Bernoulli trials is performed, with s successes and t failures and a prior given by $f(x; \alpha, \beta)$ is specified, the posterior distribution is given by $f(x; \alpha + s, \beta + t)$. This allows easy extraction of statistics and in the case of complete data, a simple closed form updating rule. These ideas are expanded upon by Castillo et al. (1997b).

2.3.2 Continuous Variables

Whilst a lot of the literature on Bayesian networks assumes that the data is multinomial, for many applications, the data supplied is continuous and so ways must be found to handle this situation. Whilst the simplest method might be to discretise the data as done by Monti and Cooper (1998), this can cause problems. However, there exist methods for representing continuous data under different assumptions. One of the first of these assumptions is that the data is normally distributed. Geiger and Heckerman (1994) use this to learn using continuous data. Taking away the normal assumption, Hofmann and Tresp (1996) use kernel density estimators to model the conditional distribution. These two methods are compared by John and Langley (1995), who show that the non-parametric approach of kernel density estimators can be useful. Another non-parametric way of estimating the conditional densities is given by Monti and Cooper (1997a), who use neural networks in this regard. They also look at the situation of hybrid networks, i.e. Bayesian networks with continuous and discrete attributes.

2.3.3 Missing Data/Hidden Variables

One large problem in learning Bayesian networks, and indeed in running any machine learning algorithm is dealing with missing data, a problem that occurs in perhaps most real-life data sets. There are generally three different missing data assumptions that can be applied to missing data. Under a missing completely at random (MCAR) assumption, the missing value mechanism depends neither on the observed data nor on the missing data. This means that the data with missing values can simply be discarded. This is an

extremely easy situation to implement, but suffers, in that there is a decreased amount of data available. Under a missing at random (MAR) assumption, the missing value mechanism depends on the observed data. This means the missing data can be estimated from the observed data. This is more complicated than the MCAR situation, but all the data gets used. And under a missing not at random (MNAR) assumption, the missing value mechanism depends on both the observed and missing data. Because of this, a model of the missing data must be supplied. This is the most complicated situation, as a model may not be readily available, or could even be unknown.

2.3.3.1 Missing at Random

One of the most widely used methods of parameter estimation with missing data is the expectation maximisation (EM) method of Dempster et al. (1977). This was first applied to learning in Bayesian networks by Lauritzen (1995). The popularity of this model probably stems from the fact that it always converges to a maximum, albeit a local one in multi-modal distributions. Extensions to this algorithm that can make it faster are given by Thiesson (1995), Bauer et al. (1997) and Neal and Hinton (1999).

As well as using EM, the gradient of the learning surface can be computed explicitly and a gradient descent applied. Russell et al. (1995) and Binder et al. (1997) apply this to the learning of parameters with possible hidden variables. They also extend this to the case of continuous nodes and dynamic Bayesian networks. Kwoh and Gillies (1996) apply the same idea, but also describe the technique of inventing hidden nodes to describe dependencies between variables. Thiesson (1997) shows an application of these ideas when prior expert information is available.

The methods given above find a local maximum of the distributions. In case a better estimate needs to be found, Monte-Carlo methods can help, such as the candidate method as used by Chickering and Heckerman (1997). Other techniques that tend to be used in structure learning might also be able to help; these are described in more detail in 2.4.11.

2.3.3.2 Missing Not at Random

When the mechanism of the missing data cannot be found from the observed data, it must be specified in some other manner. The Bound and Collapse (BC) method given by Ramoni and Sebastiani (1997a,b) can be useful in this regard. In (Ramoni and Sebastiani, 1999), they compare BC to EM and to the Gibbs sampling Monte-Carlo method and show that BC can be substantially faster. A method related to BC is the Robust Bayesian

Estimator (RBE) of Ramoni and Sebastiani (2001). Here, an assumption on the type of missing data does not need to be made. Instead, probability intervals are calculated that can be used in inference and provide a more robust estimate.

2.3.4 Miscellaneous Techniques

This section will show some techniques in learning parameters that look at specific topics.

Firstly, researchers have looked at learning parameters in causal independence models, i.e. models where causes can be assumed to be independent from each other, e.g. in noisy-OR and noisy-MAX nodes. Meek and Heckerman (1997) show how these types of nodes can be learnt using Bayesian methods, whilst Neal (1992) shows learning noisy-OR and sigmoid models using Gibbs sampling.

The simplest model of a multinomial conditional probability distribution is probably representing it as a table of values. However other representations may be possible, such as trees, that can model interactions between variables at a finer level. For example, Friedman and Goldszmidt (1996b) demonstrate simple algorithms that can learn conditional probability distributions as tables or trees, as part of an overall structure learning algorithm. In the same vein, Chickering et al. (1997a,b) show an algorithm that learns decision graphs for the CPDs as well as the network structure.

In regards to learning dynamic Bayesian networks, Ghahramani and Jordan (1997) discuss learning the parameters of a factorial hidden Markov model (and hence a dynamic Bayesian network). This is generalised to dynamic Bayesian networks and an analysis is done over many different specialisations of DBNs (Ghahramani, 1998).

Normally, updating parameters in an online setting is not a hard task, but when coupled with structure learning, there can be difficulties in knowing what data to remember. An early look at this problem is given by Buntine (1991), who describes a system of keeping possible parameters for a node in a lattice structure. Bauer et al. (1997) look at a different problem, with updating parameters in an online setting, assuming missing data.

Finally, the papers below represent some interesting ideas in parameter learning, with possible applications to structure learning. As a prelude to their structure learning method described in Section 2.4.15, Tong and Koller (2000) present an application of using active learning to estimate parameters in a Bayesian network. Also in the context of structure learning, Greiner et al. (1997) examine ways of learning CPDs dependent on the queries that will be put to the network.

2.4 Learning Bayesian Network Structures

Learning the structure of a Bayesian network can be considered a specific example of the general problem of selecting a probabilistic model that explains a given set of data. Although this is a difficult task, it is generally considered an appealing one, as constructing a structure by hand might be hard or even impossible if the dependent variables are not known by domain experts. Because of this problem, a wealth of literature has been produced that seeks to understand and provide methods of learning structure from data.

A fine example of an overview on the area was given by Buntine (1996), which although dated now, is a good reference in dealing with most of the issues that arise in the area. Heckerman (1995b) gives a more tutorial like introduction to the task, and for a gradual introduction to the area, the recent book by Neapolitan (2004) has a good look at the theory behind a lot of the techniques used. To begin with, this section will start with a look at the theory and complexity of learning Bayesian network structures and then move on to how the challenges have been addressed.

2.4.1 Learning Theory and Learning Complexity

There is a lot of theory behind the learning of Bayesian networks, most of which is rooted in statistical concepts and graph theory. Geiger et al. (2001) and Geiger (1998) look at different families of models (of which Bayesian networks are one) in the context of model selection, but a gentler introduction can be found in the pages of the books of Pearl (1988), Jensen and Nielsen (2007), Castillo et al. (1997a) and Cowell et al. (1999). From a more recent perspective, Kočka et al. (2001) investigate the important role of inclusion in learning Bayesian network structure. And whilst the theory of learning is important as a basis to why certain techniques are adopted, to many people, the issue of complexity of learning is the most immediately obvious challenge.

2.4.1.1 Complexity

Learning Bayesian network structures has been proven to be NP-hard by Chickering (1996a) and Chickering et al. (2004), whilst Dasgupta (1997) has looked at the situation where latent variables are and are not allowed. Indeed, a simple look at the number of possible DAGs for a given number of nodes will indicate the problem is hard; for 10 nodes there are 4.2×10^{18} possible DAGs. The properties of the space of DAGs have been explored by Gillispie and Perlman (2001, 2002) who look at equivalence classes of DAGs

and Steinsky (2003) who presents an efficient scheme of coding labelled DAGs.

Luckily, from the theoretical standpoint, it is possible to put bounds on various items of interest. For example, Friedman and Yakhini (1996) look at the sample complexity of structure learning and show how many samples are needed to achieve an ε -close (in terms of entropy distance) approximation, with confidence probability δ . Zuk et al. (2006) show how to calculate the number of samples needed to learn the correct structure of a Bayesian network.

Despite the complexity results, various techniques have been developed to render the search tractable. The following sections will show these in the context of the three main methods used:

- A score and search approach through the space of Bayesian network structures (Section 2.4.4);
- A constraint-based approach that uses conditional independencies identified in the data (Section 2.4.7); and
- A dynamic programming approach (Section 2.4.10).

Although the classification into three different methods is useful in differentiating their applicability, the boundaries between them are often not as clear as they may seem. E.g. the score and search approach and the dynamic programming approach are both similar in that they use scoring functions. Indeed, there is a view by Cowell (2001) that the conditional independence approach is equivalent to minimising the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) using the score and search approach.

Whilst these three approaches will be illustrated, other factors that impact the process will be mentioned. These include: partially observed models, missing data, multi-model techniques, dynamic Bayesian networks, parallel learning, on-line learning, incorporating prior knowledge into learning, large domains, continuous variables, robustness of learning, tricks to make learning faster and other problems and techniques that could be relevant.

2.4.2 Trees

One of the first pieces of work on learning structure was by Chow and Liu (1968), who described an algorithm for learning Bayesian networks structured as trees, i.e. a structure where each node has either one or zero parents. These are sometimes known as Chow-Liu trees. Their algorithm constructs the optimal second-order approximation to a

joint distribution, by finding the maximal spanning tree, where each branch is weighted according to the mutual information between the two variables. This work was built upon by Ku and Kullback (1969) who show it is a special case of a more general framework to approximating joint probability distributions.

There has continued to be research on trees as a decomposition of a joint distribution e.g. by Lucas (2002) and Friedman et al. (1997) in the context of classification. Meilă and Jaakkola (2006) show how learning tree structures in a fully Bayesian fashion can be achieved in polynomial time.

2.4.3 Polytrees

More general than trees, polytrees are an important class of Bayesian network structure. A polytree is a graph in which there are no loops, irrespective of arc direction. They are important because there exist exact algorithms that can perform inference on the polytree in polynomial time (Kim and Pearl, 1983; Peot and Shachter, 1991).

One of the earliest examples on learning polytrees from data is given by Pearl (1988), following on from work by Rebane and Pearl (1987), which uses Chow and Liu's (1968) system as a subroutine. Dasgupta (1999) gives a good look at the field and mentions the NP-hardness of the problem, whilst showing a good approximation. Other work on the area includes Geiger et al. (1990) and Acid and de Campos (1995), who show an empirical study into approximating general Bayesian networks by polytrees.

We will now turn our attention to the problem of learning a general Bayesian network structure, i.e. a DAG. This has by far received the most attention from the research community and correspondingly there are many more publications. In the sections that follow, there will be a classification of the various factors involved, but it is worthwhile to bear in mind that some ideas fall into many different camps.

2.4.4 Heuristic Algorithms

One of the most widely studied ways of learning a Bayesian network structure has been the use of so-called 'score-and-search' techniques. These algorithms comprise of:

- A search space consisting of the various allowable states of the problem, each of which represents a Bayesian network structure;
- A mechanism to encode each of the states;
- A mechanism to move from state to state in the search space; and

- A scoring function to assign a score to a state in the search space, to see how good a match is made with the sample data.

Because of the hardness of the problem, heuristic algorithms are generally used to explore the search space, the most basic of which are greedy searches. In all these frameworks, it is useful to bear in mind the work of Xiang et al. (1996), who show that single-link search cannot find all models.

2.4.4.1 Greedy Search with an Ordering on the Variables

Some of the earliest work that looked at greedy methods to learn Bayesian network structure was by Herskovits and Cooper (1991) with their Kutató system. However, the seminal paper in this area is by Cooper and Herskovits (1992), which describes the K2 system.¹ This provided a way to construct a Bayesian network structure given a data sample and an ordering of the various variables and used a Bayesian scoring criterion, which has come to be known as the K2 score.

Following on from this, Bouckaert (1993, 1994a) developed his K3 system that, like the K2 system, takes an ordering of variables and a set of data and produces a DAG. Instead of using the K2 score, he uses a scoring criterion based on the minimum description length (MDL) principle (Section 2.4.6.2). de Santana et al. (2007) have a procedure that behaves like K2, in that it needs an ordering on the variables and decides whether to add an arc from a possible parent by looking at a regression coefficient. Similar again is the work of Liu et al. (2007b) and Liu and Zhu (2007a,b), which takes an ordering of the variables and treats the problem as a feature selection one.

2.4.4.2 Greedy Search with No Ordering on the Variables

Other people that used the MDL scoring function were Lam and Bacchus (1993, 1994a) who had a best-first search algorithm and a way to incorporate domain knowledge into the problem. Suzuki (1999) also used MDL in conjunction with branch and bound. Branch and bound is a technique that has been used in many AI applications (Miguel and Shen, 2001) and that prunes the search space of definitely worse solutions, using bounds obtained from the current best solution.

One of the most important works on learning structures was by Heckerman et al. (1995), who analysed scoring functions from a Bayesian perspective and tested their techniques using a greedy learning algorithm described by Chickering et al. (1995), that

¹The name K2 is derived from the Kutató system that preceded it.

added, removed or reversed an arc from the current DAG at each step. Following on from this general technique, various researchers showed methods that seek to make learning faster and more accurate. Chickering et al. (1997a,b) show an algorithm that learns decision graphs for the CPDs at each of the nodes as part of structure learning. Steck (2000) has a search technique that alternates between the search space between DAGs and skeletons. Hwang et al. (2002) have a method to reduce the search space, whilst de Campos et al. (2002c) introduce a modified neighbourhood.

2.4.4.3 Genetic Algorithms

There has been a tremendous amount of interest in using genetic algorithms (GAs) to learn Bayesian network structures in the recent past. One of the first implementations came from (Larrañaga et al., 1996a,b), who used GAs to search over the space of orderings, whilst using K2 as a subroutine to score a particular ordering. A closely related approach comes from Hsu et al. (2002), who have the same basic idea, but hold back training data to produce a score for an ordering using importance sampling. Finally with his K2GA algorithm, Faulkner (2007) again uses a modified K2 as a subroutine for a GA.

Following on from the work of Larrañaga et al., Wong et al. (1999) introduced their minimum description length and evolutionary programming (MDLEP) system, which searches over the space of DAGs, mainly by mutating individuals. An interesting hybrid technique that combines a mixed approach of score-and-search with conditional independence testing and evolutionary programming is given by Wong et al. (2002), with their hybrid evolutionary programming (HEP) system. Following on from their previous work they introduce another system, hybrid evolutionary algorithm (HEA), again based on a hybrid approach (Wong and Leung, 2004). This is extended to deal with missing data in the *HEAm* system (Guo et al., 2006).

Myers et al. (1999a,b) compare using an evolutionary algorithm against a Markov-chain Monte-Carlo (MCMC) algorithm and also combine them to form the evolutionary MCMC (EMCMC) algorithm. Whilst this approach is focused on model selection, Wang et al. (2006) look at the problem from the perspective of model averaging with their DBN-EMC system.

Compared to normal Bayesian networks, dynamic Bayesian networks normally do not receive as much attention. Tucker and Liu's (1999) and Tucker et al. (2001) EP-Seeded-GA algorithm fills this gap with an evolutionary programming approach to learning dynamic Bayesian networks with large time lags. A more recent example of this is the genetic algorithm based on greedy search (GA-GS) algorithm of Gao et al. (2007).

Combining Many researchers have investigated combining GAs with other techniques from the machine learning library. E.g. following on from the on-line algorithm of Friedman and Goldszmidt (1997), Tian et al. (2001) have a procedure (IEMA) that combines an evolutionary algorithm and the expectation-maximisation procedure to learn structure in the context of hidden variables. Blanco et al. (2003) use techniques based on estimation of distribution algorithms (EDA), which are similar to GAs and compare them to straight GAs. Morales et al. (2004) use a fuzzy system that combines the values of different scoring criteria, whilst performing a GA search. Finally, Delaplace et al. (2006) showcase a refined GA, which includes tabu search and a dynamic mutation rate.

Representation The effective representation of population members and by extension the search space is a difficult problem that has borne much scrutiny. Most authors define their own representation and concentrate on other matters, but Novobilski (2003) is concerned with the encoding of DAGs. These issues also arise in the works of Cotta and Muruzábal (2004) and Cotta and Muruzábal (2002), who look at searching through both the space of DAGs and the space of equivalence classes of DAGs. Finally, van Dijk and Thierens (2004) and van Dijk et al. (2003a) look at the encoding of solutions so as to eliminate redundancy in the search space.

2.4.4.4 Simulated Annealing

Although implementing a search using simulated annealing (Kirkpatrick et al., 1983) should throw up no conceptual problems as it uses the framework already specified for heuristic search in Section 2.4.4, there does not seem to be much literature on the effectiveness of this approach. This is surprising, as it is very similar to a greedy search that does not always select the best neighbouring state. Instead, it picks one at random and moves to it with probability given by the scoring function of that state and how many iterations have passed. One such work that does look at this technique is by de Campos and Huete (2000a), who compare GA and SA on a search over orderings.

2.4.4.5 Particle Swarm Optimisation

Quite recently there has been work on applying discrete particle swarm optimisation (Kennedy and Eberhart, 1995, 1997) to learning Bayesian network structures. Xing-Chen et al. (2007a) and Heng et al. (2006) have applied this in the case of normal Bayesian networks and also in the case of dynamic Bayesian networks (Xing-Chen et al., 2007b). Other

approaches include that by Li et al. (2006), who use a memory binary particle swarm optimisation technique and that by Sahin and Devasia (2007) who use a distributed particle swarm optimisation approach.

2.4.4.6 Other Heuristics

There remain many other methods that have been and could be used in learning the structure of a Bayesian network. A selection of these are given here in order to complete this look at the use of heuristics.

Peng and Ding (2003) have an extension of the K2 algorithm, called K2+, that works locally on each node, eliminating any cycles obtained and repairing damage due to cycle elimination. Recognising stochasticism as a method to avoid local maxima, de Campos and Puerta (2001b) describe a randomised local search called Variable Neighbourhood Search. Additionally, de Campos et al. (2002a) apply the ant colony optimisation meta-heuristic to searching in the space of DAGs and of orderings of nodes (de Campos et al., 2002b). Their method, ACO-B, is what the main body of work of this thesis is based on and is described in more detail in Section 3.5.2. Burge and Lane (2006) describe a method based on aggregation hierarchies, that perform initial search on composite random variables. This constrains later searches using atomic random variables.

2.4.5 Searching Through the Space of Equivalence Classes

As the structure of a Bayesian network is a DAG, it is natural to use this representation as a state whilst searching through the space of possible structures. However, it has been noted that certain DAGs are similar in that they capture the same conditional independencies in their structure (Andersson et al., 1997). These Markov equivalent structures have been discussed in Section 2.1.3. Since the PDAG structure discussed in that section can represent an equivalence class of DAG structures, it is very useful in representing states of searches. The space of these searches can be known as E-space, as opposed to the B-space of DAG based search (Chickering, 2002a). More information on these topics can be found in Lauritzen and Wermuth (1989) and Whittaker (1990).

The main work of this thesis involves learning Bayesian networks by searching through the space of equivalence classes of structures. As such it is very relevant to the work that will be described later in Section 4.3.

2.4.5.1 Search Procedures

Whilst the properties of PDAGs have been known for some time before, algorithms that would learn them from data, in a manner similar to score-and-search procedures to find DAGs, would not appear until later. One of the first was by Spirtes and Meek (1995) who describe a two-phase greedy Bayesian pattern search (GBPS) algorithm and then combine it with the independence-based PC algorithm (Spirtes and Glymour, 1990). This work relies on a procedure to turn a PDAG \mathcal{P} into a DAG in the equivalence class represented by \mathcal{P} (known as extending \mathcal{P}). Such procedures are described by Meek (1995), Verma and Pearl (1992) and Dor and Tarsi (1992).

Another early work is by Chickering (1996b), who describes a method that uses certain operators to modify a PDAG \mathcal{P} and then extends \mathcal{P} to \mathcal{G} to check if the move is valid and to score it. It then turns \mathcal{G} back into a PDAG and repeats, using a method such as those by Meek (1995) or Chickering (1995).

A problem with these procedures was that they were often very inefficient, with numerous extensions and multiple scores being required at each move. These problems were addressed by Munteanu and Cau (2000) and Munteanu and Bendou (2001) with their EQ framework, who showed how to locally check if a particular move was valid and if so, what score that would provide. However, the various operators given were shown to be incorrect. Kočka and Castelo (2001) tried to limit the problem inherent with searching in the space of DAGs, by including a procedure that would move between DAGs in the same equivalence class. However, it was the paper by Chickering (2002a) that put a firm foundation on using equivalence classes as states in a search-based procedure. Whilst similar to the procedure of Munteanu and Bendou (2001), he proved the correctness of the various operators introduced and enabled search in E-space to be competitive with that in B-space. Note that Perlman (2001) and Castelo and Perlman (2002) also did work on this problem.

After this, Chickering (2002b) designed another algorithm that searches in E-space. This one, called greedy equivalent search (GES), is a two-phase algorithm that, in the limit of a large sample size, identifies a perfect map of the generative distribution. That is, if the probability distribution implied by the data admits a DAG representation of it, then GES will find it in the limit of a large sample size. This work was expanded on by Chickering and Meek (2002), who provide different optimality guarantees for more realistic assumptions. Nielsen et al. (2003) also built on GES by introducing an algorithm called k -greedy equivalence search (KES), which is essentially a randomised version

of GES, to help escape local optima in the search space. Quite recently, Borchani et al. (2006) developed the GES-EM algorithm for utilising GES with missing data, using the expectation maximisation procedure.

Following on from this work, Castelo and Kočka (2003) show a more general way of looking at the search problem and illustrate certain conditions that operators on the search space should obey, to avoid local maxima. They then introduce the hill-climber Monte-Carlo algorithm (HCMC) that uses the ideas developed in this paper.

To finish off this section on search algorithms, various hybrid and other methodologies will be mentioned. Acid and de Campos (2003) develop a representation that borders the representational ability of DAGs and CPDAGs. They named these restricted PDAGs (RPDAGs) and present various operators that can be used to manipulate them. Cotta and Muruzábal (2004) and Muruzábal and Cotta (2004) present an evolutionary programming approach called EPQ, that uses equivalence classes of structures as its population members. Finally, Jia et al. (2007) show a hybrid algorithm using a conditional independence approach and score-and-search to learn an equivalence class.

2.4.6 Scoring Functions

When performing a score-and-search procedure a scoring criterion must be specified that somehow gives a good score when a structure matches the data ‘well’ – the better the match, the higher the score. Of course, given this ambiguous problem, diverse scoring criteria have been invented that involve various assumptions and different definitions of a better match. Perhaps one of the simplest criteria – the maximum likelihood estimator – will, in general, return the complete graph, as this is the one with the most parameters. Therefore, most scoring criteria consist of two parts – one that rewards a better match of the data to the structure and one that rewards a simpler structure. Examples of these are the Bayesian Dirichlet (BD) criterion, Bayesian information criterion (BIC), Akaike information criterion (AIC), minimum description length (MDL) and minimum message length (MML). These and various other criteria will be discussed below.

One of the more desirable properties of a scoring criterion is decomposability, whereby the score of a particular structure can be obtained by the score for each node given its parents. Most of the scoring functions known have this property. Malvestuto (1991) has some early work on this.

A comparison of MDL, BDeu (BD with $N'_{ijk} = N'/r_i q_i$) and K2 (BD with $N'_{ijk} = 1$) is given in Shaughnessy and Livingston (2005). Note that in this paper, they confuse

BDeu with K2 and the WEKA Bayesian method with BDeu. Another thorough comparison between the BIC, Cheeseman-Stutz approximation (Cheeseman and Stutz, 1996), Laplace approximation and Gibbs sampling approach was conducted by Chickering and Heckerman (1997)

2.4.6.1 Bayesian Dirichlet Scoring Functions

One of the first expositions of a Bayesian scoring criterion in learning Bayesian network structure was given by Cooper and Herskovits (1992) as part of their K2 algorithm. As presented by them, the function for a given structure B_s and data set D was

$$P(B_s, D) = P(B_s) \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!,$$

where there are n variables, q_i parent configurations of variable i , r_i is the number of values variable i can take, N_{ijk} is the number of times variable i took on value k , with parent configuration j and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. Looking at this equation it can be seen that $P(B_s)$ is the prior probability of structure B_s and the rest of the expression is the likelihood of the structure, given the data.

In their paper, Heckerman et al. (1995) generalise the above equation and place it on a sound theoretical footing. In their form,

$$P(B_s, D) = P(B_s) \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})},$$

where variables have the same meaning as before and N'_{ijk} are exponents that specify the user's prior knowledge about configuration ijk ; the higher N'_{ijk} , the more the user thinks that configuration is likely. Meanwhile, $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$. They called this equation the Bayesian Dirichlet or BD metric. It can be noted that $\Gamma(x) = (x-1)!$ for natural numbers, which immediately shows the similarity to the equation of Cooper and Herskovits (1992). Indeed, with $N'_{ijk} = 1$, the formula is exactly the same as the K2 formula. Using an assumption known as likelihood equivalence, which says that equivalent structures receive the same score, they constrain the values that N'_{ijk} can take on and provide a method to calculate them. With this method, the user provides a prior network structure and an equivalent sample size N' that says how confident they are in it. With these constraints, the scoring criterion is named BDe (Bayesian Dirichlet with likelihood equivalence). With a further constraint, such that all configurations are as likely as each

other, $N'_{ijk} = N'/r_{iq_i}$ and the criterion is named BDeu (Bayesian Dirichlet with likelihood equivalence and a uniform joint distribution).

The BDeu scoring function is used in this thesis as the function through which the quality of changes to an equivalence class of Bayesian network structures is quantified through the operators described in Section 4.3.

2.4.6.2 Minimum Description Length

Minimum description length (MDL) is a statistical principle, which says that the best hypothesis that describes data, is the one that leads to the largest compression of that data (including the hypothesis) (Rissanen, 1978). One of the first people to use the MDL principle in constructing Bayesian networks was Bouckaert (1993, 1994a). Here he provides an explanation and justification of using this measure and shows how it behaves like the K2 criterion for large sample sizes. A more in-depth comparison of the MDL and K2 criterion is given in (Bouckaert, 1994b).

At around the same time, Suzuki (1993) and Lam and Bacchus (1994a) also made use of the MDL principle and presented a way to incorporate domain knowledge into their algorithm (Lam and Bacchus, 1993). This domain knowledge can come in the form of direct causation information (X is a direct cause of Y) and a partial ordering of the variables. Suzuki (1999) looks into previous work using MDL and presents a branch-and-bound algorithm. Finally Cruz-Ramírez et al. (2006) give a comparison of MDL and BIC. In this work BIC is defined as

$$\text{BIC}(B_s, D) = -\log P(D|\hat{\Theta}, B_s) + d/2 \log n,$$

where $\hat{\Theta}$ are the maximum likelihood parameters for B_s , d is the number of free parameters in the model and n is the sample size. Following this, MDL is defined as

$$\text{MDL}(B_s, D) = -\log P(D|\hat{\Theta}, B_s) + d/2 \log n + C_k,$$

where $C_k = \sum_{i=1}^k (1 + |Pa_{x_i}|) \log k$ and $|Pa_{x_i}|$ is the size of the parent set of variable x_i . Hence according to Cruz-Ramírez et al., MDL is BIC with an extra penalty for model complexity. Note that some authors use the term MDL to mean the formula indicated by BIC above, so care must be taken to see exactly what is being used.

2.4.6.3 Minimum Message Length

A newer scoring criterion for Bayesian networks, based on the work of Wallace and Boulton (1968), is one that computes the minimum message length (MML) of a structure,

its parameters and data (Wallace et al., 1996). This criterion is similar to the MDL one in that it penalises overly complex models (which will produce a longer message) and rewards goodness of data fit (which will produce a shorter message). However, instead of looking at the problem from a compression point of view, it views it as the problem of sending the smallest possible message between a transmitter and receiver. It has been used to learn linear causal models by Neil et al. (1999) and by Wallace and Korb (1999) using their causal minimum message length (CaMML) method, and more recently by Li et al. (2002). An up-to-date look at this method is given by Korb and Nicholson (2004). O'Donnell et al. (2006b) builds on the work of Wallace and Korb (1999) by learning Bayesian networks with many types of local interactions at each node.

2.4.6.4 Other

Whilst the scoring criteria given above are the most widely used in the field, researchers have been examining other methods of ranking a structure given some data. For example, Kayaalp and Cooper (2002) introduce a scoring metric called Global Uniform (GU) that is based on a particular form of default parameter prior. They then compare it against the BDeu and K2 scoring criteria. de Campos (2006) presents a scoring method called mutual information tests (MIT) that has an information theoretic basis and performs well compared to K2, BDeu, BIC and the PC algorithm of Spirtes and Glymour (1990).

An interesting approach with similarities to boosting etc. is that by Elidan et al. (2002) who reweight data, so that the scoring function can change. This can help escape local maxima. Castelo and Perlman (2002) demonstrate a scoring criterion that can be applied directly to equivalence classes of Bayesian network structures, without extending it to DAG form.

2.4.7 Finding Structure using Conditional Independencies

In learning the structure of a Bayesian network from data, there are often said to be two main methods – search through the space of possible structures and using conditional independencies (CIs) obtained from statistical tests on the data. In this section the latter will be focused on. However, it is worth noting that Cowell (2001) has drawn parallels between the two techniques, so they might not be as different as they seem at first glance. Perhaps *the* book for learning using CIs is the one by Spirtes et al. (2000), which contains most of the early theory and results, and looks at the problem from a causal perspective.

Some of the first work on using CIs to obtain structure came from around the

early 1990s. Geiger et al. (1990) developed an algorithm to recover polytrees from an independence oracle, i.e. an oracle which can say whether two variables are independent given another. At the same time Fung and Crawford (1990) developed the Constructor system that learnt Markov networks, i.e. undirected graphical models, that encode CIs in a manner similar to Bayesian networks. In this work, they mention using the χ^2 statistical test to obtain the needed CIs.

One of the first systems that recovered a DAG from independence data was the SGS algorithm by Spirtes et al. (2000). However, this was quite inefficient, as each pair of variables required tests involving every subset of the remaining variables, which is an exponential operation. Therefore, numerous variations on the theme sprung up. Perhaps the best known of these is the PC algorithm by Spirtes and Glymour (1990), which is faster than SGS, but can produce errors in removing arcs. These arise because PC only tests for d-separation between nodes X and Y in a DAG, using subsets of neighbours of X and Y . A modification of the PC algorithm that decreases the amount of CI tests needed is called PC* (Spirtes et al., 2000), and recently, applicability of PC to high dimensional data is shown by Kalisch and Bühlmann (2007).

A variant of the SGS algorithm, proposed by Pearl and Verma (1991) and Verma and Pearl (1991), differs from previous approaches in that it first generates an undirected graph that models dependencies between variables, as opposed to using the complete undirected graph in the SGS and PC manner. This algorithm, called Inductive Causation (IC), also takes into account latent variables, and returns a graph with undirected, unidirected and bidirected arcs. Taking inspiration from the approach of IC in starting with the undirected independence graph, Spirtes et al. (2000) proposed changing the PC algorithm to start in the same way. They called this modified algorithm Independence Graph (IG). Again, with an approach based on IC, Verma and Pearl (1992) looked at an algorithm that took a more global view and constructed a DAG, as opposed to the local view of IC.

Following on from the work of Verma and Pearl, Spirtes et al. (2000) described two algorithms that took on the idea of identifying latent variables. These were known as the Causal Inference (CI) and Fast Causal Inference (FCI) algorithms and Spirtes et al. (1995) show how they can also work in the presence of, and identify, selection bias.

After the large body of work produced by Spirtes et al. and Verma and Pearl, various other authors examined ways to learn causal explanations of data in the context of DAGs. These are refinements of the more general approaches discussed above and were often developed for simplified situations. Cooper (1997) described an algorithm called Local Causal Discovery (LCD) based on PC and FCI, that while less general in its applicability,

runs in polynomial time in the worst case. de Campos (1998) looked at representing CI statements using polytrees and developed algorithms for this purpose. de Campos and Huete (1997) also make a simplifying assumption with their CH1 and CH2 algorithms that can efficiently find simple graphs; i.e. DAGs where every pair of nodes with a child are not connected, nor have a common ancestor. Various authors also advanced systems that allow background knowledge to be given. E.g. Meek (1995) showed a method for learning a CPDAG using methods inspired by Spirtes et al. This was done in the context of background knowledge in the form of mandatory and disallowed causal effects. This paper also has early work on finding DAGs from CPDAGs. Cheng et al. (1997) have a system that takes an ordering of the variables as background knowledge, as do de Campos and Huete (2000b), who describe a method that avoids making many high order CI tests.

An interesting modification to the standard CI type algorithm is given by Margaritis and Thrun (2000) with their GS algorithm, which shows how to limit the number of CI tests between two nodes by only using nodes in their Markov boundaries, which are calculated beforehand. Another interesting approach is that of Cheng et al. (2002) and their Three Phase Dependency Analysis (TPDA), who describe how an n variable DAG can be learnt in $O(n^4)$ CI tests. This was also implemented in parallel by Gou et al. (2007). However, TPDA relies on an assumption known as monotone DAG-faithfulness, and it has been shown by Chickering and Meek (2006) to be incompatible with the faithfulness assumption. They show that the optimality guarantee provided by Cheng et al. only applies in very specific situations, where there are already faster algorithms.

Some recent work on reducing computation time and errors in reconstructing networks is given by Yehezkel and Lerner (2006). They propose an algorithm known as recursive autonomy identification (RAI), that recursively performs CI tests, edge directions and structure decomposition, with higher order CI tests for smaller structures.

With all the algorithms given above, access to fast CI tests is crucial. The normal tests used are the χ^2 and G tests (Spirtes et al., 2000, pp.93–95), but work has been done on a new test by Dash and Druzdzel (2003). Other results concerning d-separations are given by Acid and de Campos (1996a,b), who use them in their hybrid BENEDICT system (Acid and de Campos, 1996c).

Finally, Schulte et al. (2007) provide a look at learning BN structures using CI tests in a paradigm that approaches the task using Gold's learning paradigm (Gold, 1967).

2.4.8 Hybrid Search Strategies

Both the score-and-search and conditional independence testing methods have their advantages. Score-and-search typically works better with less data than CI testing and with probability distributions that admit dense graphs. They also allow probability distributions over models to be easily represented and have better mechanisms for dealing with missing data. On the other hand CI testing methods work well with sparse graphs, are generally quick and have good ways of finding hidden common causes and selection bias.

Because both methods have advantages inherent in them, researchers have tried to find ways to use the good points of both in hybrid methods. Below are some of the ideas that researchers have been looking at.

One of the first hybrid algorithms in the area was by Singh and Valtorta (1993, 1995). Here, they construct a total ordering of the variables using CI tests and then use this ordering as input to the K2 algorithm to learn the structure. This approach is followed by Provan and Singh (1996), with their CB system. Whilst similar to the first work, the later approach employs an initial feature selection phase. Acid and de Campos (1996c, 2001) take a different route, by measuring the difference in independencies between a candidate graph and the data, using the Kullback-Leibler cross-entropy. They named this algorithm BENEDICT, a refinement of which is found in their BENEDICT-*dsep* system (Acid and Campos, 2000).

An approach that combines the main features of both techniques is given by the EGS algorithm of Dash and Druzdzel (1999), who use PC to obtain an initial guess of a PDAG, extend it to a DAG and then perform a greedy search. de Campos et al. (2003) do the opposite with their IMAPR algorithm; they perform an initial greedy search with random restart and then use CI tests to add and delete arcs from the obtained DAG. Whilst the work of Friedman et al. (1999c) is slightly similar, it stops short of providing a full structure in the initial phase. Instead, it uses CI to find good candidate parents and hence limit the size of the search in later stages. This algorithm, called sparse candidate (SC) is very useful in increasing the speed of search procedures, without unduly damaging the score. It has also been built upon by later authors. Brown et al. (2004) and Tsamardinos et al. (2006) describe the max-min hill-climbing (MMHC) algorithm that differs mainly from the SC algorithm in the way its candidate parent sets are generated, that is, by the MMPC algorithm (Aliferis and Tsamardinos, 2002).

Insofar as their first phase builds a conceptual skeleton upon which arcs are directed,

the work of van Dijk et al. (2003b) is quite similar. A polynomial version of the initial phase of MMHC is given by Brown et al. (2005). Known as Polynomial Max-Min Skeleton (PMMS), it is compared to TPDA. Like the latter, it relies on assumptions that restrict the structure of networks that can be found. Another algorithm relying on the monotone-DAG-faithfulness assumption and CI testing to generate an initial skeleton is given by Wang et al. (2007). They tested their approach in the regulatory gene domain.

Lately, MMHC has been extended to cope with very large domains (in the order of thousands of variables) by Nägele et al. (2007). Their approach focuses on learning substructures around each variable. An extension that focuses on equivalence classes of structures is given by Jia et al. (2007) who first learn a skeleton of a graph using CI tests and then try to identify all the v-structures.

Other hybrid algorithms can use CI tests to lesser extent. E.g. in HEP, discussed in Section 2.4.4.3, CI tests are used to disallow certain arcs in generated structures (Wong et al., 2002). In (Huang et al., 2005), part of the algorithm presented uses CI tests to remove edges from an undirected graph. This graph will later be changed to a directed graph.

2.4.9 Searching over Orderings

Score-and-search algorithms for finding Bayesian network structures generally search over the space of DAGs and in some cases, the space of equivalence classes of DAGs. However in some cases, other spaces can be used. One that has received some attention is searching in the space of orderings.

An early paper on this subject was given by Larrañaga et al. (1996b) as seen in Section 2.4.4.3. Acid et al. (2001) use an approach similar to Singh and Valtorta (1995) as seen in Section 2.4.8, by using CI tests. However, instead of learning a definite ordering, a search is performed to preserve as many of the CIs as possible. Teyssier and Koller (2005) reported results using a true local search algorithm that proceeded by swapping the position of two adjacent variables in an ordering and performing a greedy search. To score an ordering they used the work of Friedman and Koller (2000, 2003) who showed how to efficiently score an ordering in closed form where each node has a bounded number of parents. This paper also showed how to compute the probability of a structural feature given data and as such is important by itself.

2.4.10 Dynamic Programming

Besides the two major techniques of structure learning that have been discussed, there exists a third method that is similar to the score-and-search approach, but does not have the search aspect. These methods use dynamic programming to compute optimal models for a small set of variables and in some cases combine these models. Note that small in this context is in the region of 25, whereas with an exhaustive enumeration, it would be impossible to score all models with numbers of variables greater than 6 or 7.

One of the first uses of dynamic programming in this way was by Ott et al. (2004). In this, an algorithm for finding the optimal model is given and its correctness proved. Later, Ott and Miyano (2003) show the technique can be used for arbitrary sized Bayesian networks by limiting the number of possible parents and by clustering, with possible input by an expert. At around the same time, Koivisto and Sood (2004) came up with a very similar procedure, but with a somewhat more in-depth analysis of the problem. They approached this from the perspective of trying to compute the posterior probability of a subnetwork in the spirit of Friedman and Koller (2000, 2003). Koivisto (2006) expanded on this with a faster method to compute the posterior probability of all edges in $O(n^2)$. Building on the earlier work of Koivisto and Sood (2004), Singh and Moore (2005) introduce a similar method that optimises a different form of equation, with some advantages and disadvantages. These are that their algorithm has a simpler structure and less of a memory requirement, but can be slower if there are constraints on the number of allowed incoming arcs to a node. This procedure is again approached in a different manner by Silander and Myllymaki (2006) who present a less complicated algorithm that is feasible for structures of up to 32 variables. As opposed to the methods of Koivisto and Sood and Silander and Myllymaki, their algorithm is conceptually simpler and scales better. Dojer (2006) has a variation that works on single variables and requires prior information so that the acyclicity of a graph does not have to be checked. Finally, Eaton and Murphy (2007b) apply the Koivisto and Sood technique to experimental data with possibly uncertain interventions and also combine the technique with MCMC to solve certain problems with the exact DP method (Eaton and Murphy, 2007a).

2.4.11 Missing Data and Partially Observed Data

Like the parameter learning case, learning Bayesian network structures with missing data significantly complicates matters. This section will investigate methods to learn in these circumstances and also in the circumstance that the data may be partially observed,

i.e. there may be variables that can help to explain the distribution of observed data, but there is no data for those variables. This can be seen as missing data where *all* the data is missing for certain variables.

One of the earliest papers on handling missing data whilst learning the structure of Bayesian networks, is the work of Cooper and Herskovits (1992), where they also present their K2 algorithm and scoring function. They show how to use the law of total probability to sum over all possible combinations of missing data. However, this is exponential in the number of missing items. This can also be used in the case of hidden variables, where all the data of a variable is assumed missing, but this is of course exponential in the number of data items. There is also the problem of knowing how many hidden variables there might be and what number of values they can take on. Cooper (1995) solves some of these problems by showing a hidden variable method that is polynomial (though perhaps to a high degree) in the number of data cases. He also provides a method of handling multiple hidden variables, though identifying them is still hard, with CI testing methods playing a useful role here.

Though the methods given above are quite general, they are still computationally intensive. Geiger et al. (1996) showed how the BIC scoring function could be used to score structures with missing data and hidden variables. In this, the maximum likelihood parameters can be estimated by using one of the approaches used in parameter learning with missing data, e.g. EM or a gradient based approach. Ramoni and Sebastiani (1997a) applied the BC method as shown in Section 2.3.3.2 to learning a structure with data that is missing not at random.

One of best known algorithms to learn Bayesian network structures in the presence of missing data or hidden variables is the structural EM (SEM) algorithm (Friedman, 1998). Starting with early work defining the MS-EM and AMS-EM systems, Friedman (1997) then went on to introduce the SEM system (Friedman, 1998). This interleaved model selection and the EM algorithm (to estimate parameters) and was proven to converge to a maximum. A generalisation of this approach was presented by Beal and Ghahramani (2003) who used a variational Bayesian EM algorithm. The method has also been specialised by Leray and François (2005) who use it in the space of trees with their MWST-EM algorithm, either as a good enough solution or a starting point for a more complex approach. Finally, Borchani et al. (2006) introduce GES-EM, which extends GES (Chickering, 2002b) as shown in Section 2.4.5.1 to deal with missing data.

Stochastic Algorithms A problem with EM based methods is that in most cases the maximum found is local. One way around this is to use stochastic procedures, such as hill-climbing with random restarts. A Monte-Carlo approach is shown by Chickering and Heckerman (1997). This important paper also shows the effectiveness of large sample approximations, i.e. scoring functions that are approximations to the Bayesian score and that converge to it in the limit of a large number of samples. Myers et al. (1999a,b) show using MCMC and an evolutionary algorithm to avoid the local maxima. An approach that is based on the TPDA algorithm of Cheng et al. (2002) as shown in Section 2.4.7, is given by Tian et al. (2003) and their EMI system. This is basically the TPDA algorithm augmented to use incomplete data. The EMI method is combined with a score-and-search approach by Tian et al. (2007).

These methods often work well for missing data and are normally easily applicable for hidden variables. However, a hard problem is knowing how many hidden variables to use and their cardinality. One solution to the first problem, already discussed, is to use a CI testing algorithm to suggest likely variables and locations. Another simple one is simply to add hidden variables one by one. Elidan et al. (2001) have a more sophisticated approach that looks at cliques in the structure. A solution to the second problem, that of finding the cardinality of hidden variables, is looked at by Elidan and Friedman (2001).

2.4.12 Model Averaging

Learning Bayesian network structures normally means the selection of a single structure. In itself, this can be a useful procedure, e.g. by presenting the DAG to a domain expert to suggest new relationships. Otherwise, parameters can be learnt and inference performed. However, a problem with this approach is seen when there is not much data. In this case, no one model rises high above the rest and the selection can be somewhat arbitrary, with a corresponding lack of confidence in the structure. One way around this is to have the learning procedure return multiple models instead of a single one. This could range from a small collection of the most likely to the complete space. These models can then be weighted by their probability when inference is being performed. A good introduction to these ideas, that looks at model averaging in general, can be found in (Hoeting et al., 1999).

One of the earliest algorithms used to find Bayesian network structures to average over, was provided by Madigan and Raftery (1994) with their Occam's window principle. This provides a small number of models that are not too similar but have good predictive

power. However, the main interest in early algorithms focused on a stochastic method devised by Madigan et al. (1993, 1995) to average across models using Markov chain Monte-Carlo model composition (MC^3). This method defines a Markov chain across the space of models and proceeds from model to model, computing the quantity of interest at each step and averaging over the results. These ideas are extended to equivalence classes of DAGs by Madigan et al. (1996). Giudici et al. (1999) provide a more efficient procedure to sample the chain, whilst Giudici and Castelo (2003) extend MC^3 by using different moves in the state space and provide an analysis of the distribution of various domains. Riggelsen and Feelders (2005) extend MC^3 to incomplete data with their eMC^4 algorithm.

As well as MC^3 based approaches, other systems have been developed to perform the same task. Thiesson et al. (1998a,b) describe an approach to learning what they call mixtures of Bayesian networks and mixtures of DAGs. Whilst seemingly oblivious to the work of Madigan et al., their model appears quite similar.

A related approach to Monte-Carlo algorithms across the space of DAGs is one of Friedman and Koller (2003). The difference with their work is that they average across the space of orderings of variables. This is possible, due to a fast closed form expression for the likelihood of an order, that they provide. Following on from this, Dash and Cooper (2004) show a method to average over models quickly and a procedure to find a single network that is equivalent to averaging. In fact this last idea has been implemented by various authors. Kim and Cho (2006) have a method to merge multiple Bayesian networks into a single model using an evolutionary algorithm. Gou et al. (2007) also have a method called parallel TPDA (P-TPDA) that uses TPDA as seen in Section 2.4.7 in parallel on different data sets and combines the resulting DAGs. Finally, Liu et al. (2007a) learn structures using a CI testing method and then combine the resulting DAGs into a single DAG.

2.4.13 Dynamic Bayesian Networks

The first authors to look at structure learning of dynamic Bayesian networks were Friedman et al. (1998) who broke the problem down into learning a prior network which provided initial conditions and a transition network which specified how variables behave from state to state. This problem was analysed from the point of view of both complete and incomplete data. Boyen et al. (1999) looked at a way of using SEM in learning DBNs and in particular found a novel approach to detecting hidden variables in dynamic sys-

tems. This is done by detecting non-Markovian correlations, i.e. correlations between variables that are separated by one or more time steps. Murphy and Mian (1999) show how DBNs subsume many other dynamic models into a general framework and look at the various tasks that need to be done to learn a DBN.

Whilst much of the work on static BNs can be applied to DBNs (as when expanded they *are* static BNs), sometimes there are techniques that can take advantage of DBNs' unique structure. Such is the case with Tucker and Liu (1999) and Tucker et al. (2001) who show an evolutionary programming approach to learning DBN structure. They also propose using hidden variables to model the change in dependencies over time (Tucker and Liu, 2004). Other authors have also proposed using metaheuristics in learning DBN structure; e.g. Xing-Chen et al. (2007b) show an implementation of particle swarm optimisation for this task and Gao et al. (2007) also use a genetic algorithm. And finally, Jonsson and Barto (2007) use active learning, as might be used by agents in a reinforcement learning setting.

Dynamic Bayesian networks are used in this thesis in order to model the circadian clock of the plant *Arabidopsis Thaliana*. Indeed, the learning scheme shown in Chapter 7 is based on the method of Friedman et al. (1998).

2.4.14 Parallel Learning

Since learning Bayesian network structures is a computationally hard task, many attempts have been made to speed it up. Most of these have been algorithmically based, but there have been efforts to parallelise the problem so it can be tackled by multiple computing resources. To a large extent, algorithms based on the score-and-search paradigm have, as a bottleneck, finding the sufficient statistics needed. In general, when evaluating different neighbouring states, each of them could be evaluated in parallel, which at a low level means scoring functions can be evaluated in parallel, thereby giving an opportunity for the bottleneck to be alleviated.

However, there have been some algorithms that have been structured such that parallelism takes a large part in their operation. Xiang and Chu (1999) showcase an algorithm that looks ahead multiple steps and hence is quite computationally intensive. However, they show how it can be decomposed into separate chunks. Mondragón-Becerra et al. (2006) show a fairly simple implementation of the above ideas, but a more interesting application is that by Yu et al. (2007), who show a method to parallelise SEM inside the EM part of the algorithm. It does this by performing the E (expectation) step on each

sample in parallel. Finally, an application using the CI testing paradigm is given by Gou et al. (2007) who perform TPDA in parallel and combine the results.

2.4.15 On-line Learning

Generally, learning a Bayesian network operates as a batch process—a block of data is given to an algorithm which learns a structure and the parameters for that structure. However, sometimes data is continuously being supplied to a system, and it could be useful to be able to learn from that. It is normally a fairly easy job to update the parameters of a system, given a single datum. However in the case of learning structures, it is not as simple. An early paper on refining both structure and parameters was given by Buntine (1991), who assumed an ordering on variables and stored counts on a parent lattice at each node. Lam and Bacchus (1994b) and Lam (1998) have a method to refine the structure of a BN given new data, that can incorporate a trade off between the old network and the new data. It does this by learning a partial network from the new data and uses this to improve the old network. Friedman and Goldszmidt (1997) provide a method that trades off between accuracy and storage, by only storing a certain number of past observations with which to refine the structure. This idea is expanded upon by Tian et al. (2001) in their IEMA system, who examine it in the context of hidden variables and introduce an evolutionary algorithm and EM into the procedure. Finally, Tong and Koller (2001) look at the problem from the perspective of active learning, i.e. where a learning system is allowed to intervene in its environment.

2.4.16 Incorporating Prior Knowledge

Allowing an expert to specify knowledge that can be used in a learning system is a fundamental task that can be extremely useful in situations with a low amount of data. However the learning data and expert knowledge are often in quite different forms and it can be difficult in bringing both together. With Bayesian networks, many types of background knowledge an expert can provide have already been seen in this chapter. These include an ordering of variables (total or partial), a prior network, prior equivalent sample size etc. Being able to use these is normally dependent on the algorithm in question, though score-and-search methods that are Bayesian normally require being somehow able to specify a prior distribution. These can be the forms already seen, or others which will be discussed in the papers looked at below.

One of these forms to specify a prior distribution uses ‘imaginary data’, elicited from

a domain expert as shown by Madigan et al. (1994). This makes the expert come up with typical cases and uses this database to update uniform priors to become the priors for the start of learning. Whilst specifying variable ordering as the prior knowledge in their system, Sarkar and Murthy (1996) also look at other knowledge that can be specified, e.g. by declaring variables to be cause or evidence nodes or by explicitly declaring conditional independencies across variables. Another prior elicitation method that takes an ordering of variables, is that of Castelo and Siebes (2000). However, they also take a subjective probability, that consists of the probability of a variable being another variable's parent, for all pairs consistent with the ordering. A discussion of the different types of prior knowledge that may be supplied is given by O'Donnell et al. (2006a), from specifying a full structure to indicating a correlation between nodes.

However, some of these techniques can be hard or impossible for an expert to specify, e.g. a structure over a domain that the expert simply does not know. Mascherini and Stefanini (2007) study this problem and specify a means to extract weak information from an expert, i.e. information about parts of the domain, e.g. local features, ordering of some variables, degree of connectivity etc. Another author who looks to impose local expert knowledge on a model is Thiesson (1997). In this case the prior information is defined in terms of a much more general class of models than Bayesian networks, recursive exponential models. These can be seen as regular Bayesian networks, where the local distributions are parametrised by members of the exponential family and experts can give information in the form of imprecise probabilities.

It can be difficult to find out the effect the prior information has on learning, so the study by Neil and Korb (1999) is useful in comparing two types of prior knowledge – a uniform prior over all orderings and a uniform prior over all structures with the same arc density. Another study based on three different types of prior information is given by de Campos and Castellano (2007). These types are the existence of edges, absence of edges and ordering of variables. Mansinghka et al. (2006) also look at non-expert supplied priors. With their system, variables are separated into different types or classes and prior probabilities given between the different classes.

2.4.17 Large Domains

Traditionally, structure learning algorithms for Bayesian networks had size limits in the hundreds of variables. However, applications such as genomics often have data sets with thousands or more features. Therefore, recent research has looked at ways to handle

these very wide sets. One early system was the MMBN algorithm by Tsamardinos et al. (2003c), that later developed into the MMHC system (Tsamardinos et al., 2006). This has been tested on domains with tens of thousands of variables. Another approach by Goldenberg and Moore (2004) is the SBNS algorithm, that proposes using Frequent Sets and exploiting the local structure of cached sufficient statistics. Finally, Nägele et al. (2007) have a method similar to MMHC that firstly learns a skeleton and then the substructures around each variable, with a final combination into a DAG.

2.4.18 Continuous Variables

Most Bayesian network theory is developed for the multinomial case, i.e. discrete variables with a bounded cardinality. However, in many applications, data is supplied in continuous form. One way to handle this is discretisation, or turning the continuous data into multinomial data. However, the process of discretisation can lead to errors, depending on how it was achieved. Therefore, researchers have tried to find ways to learn with continuous variables directly.

An early work on learning with continuous variables is by Geiger and Heckerman (1994), who assume the data is drawn from a multinomial Gaussian and develop scoring criteria for the case of networks with all continuous and a mixture of continuous and discrete nodes. John and Langley (1995) drop the assumption of normality and instead use non-parametric density estimators, specifically Gaussian kernels. The same approach is followed by Hofmann and Tresp (1996). Bach and Jordan (2003) also use kernels, of the Mercer variety. Slightly different is the work of Monti and Cooper (1997a,b) who use neural networks to represent the density function. A different approach, based on the CI testing paradigm is used by Margaritis (2004). In the discrete case, the χ^2 test is normally used, but this cannot be used for the continuous case. He develops a non-parametric CI test that does not rely on the variables being distributed according to a given model. This test can be used as input to any of the CI based algorithms of Section 2.4.7.

2.4.18.1 Discretisation

Some authors have proposed a different way of learning with continuous variables, that involves a discretisation stage as part of a learning algorithm. Friedman and Goldszmidt (1996a) are one of the first to do this with a modified MDL score that chooses the discretisation thresholds. Monti and Cooper (1998) also have a discretisation strategy that changes as the learning algorithm progresses. This strategy is based on the Bayesian

scoring principle, that depends on the data and the network structure. Steck and Jaakkola (2003) show that the discretisation policy can affect the structure of the graph learnt and present a scoring function that efficiently discretises data, in sequence with learning the structure.

2.4.18.2 Other Topics

It is not just model selection that researchers have concentrated on. Giudici and Green (1999) look at using MCMC across the space of structures. Imoto et al. (2002) also include an MCMC simulation of their method, based on non-parametric regression. Finally, Bøttcher (2004) looks at learning conditional Gaussian networks and also dynamic Bayesian networks with mixed variables.

2.4.19 Robustness

When the size of the sample is small, small changes to the data can produce large changes to the learnt structure. If the Bayesian network is to be used in a production environment, or to provide evidence of a dependency between variables, it is very useful if an idea of the robustness can be found. This can help decide how much confidence to place in the network. Early confidence measuring research by Friedman et al. (1999a,b) used the Bootstrap in order to find a degree of reliability of certain features in the learnt DAG, e.g. the existence of an edge, the Markov blanket of a node or the ordering of variables. Holness (2007) also examines the confidence in learning structural features, in this case causal associations between variables. Steck and Jaakkola (2002) look at robustness from a different angle and investigate the sensitivity of the 'equivalent sample size' as used in Bayesian scoring criteria. They show that a small equivalent sample size can surprisingly lead to a strong regularisation of the graph structure, i.e. the graph structure will be sparse. The work of Silander et al. (2007) is very similar in this regard; they investigate the sensitivity of the learnt structure to the value of the 'equivalent sample size'.

2.4.20 Acceleration Techniques

Since learning Bayesian network structures is in general a hard problem, various techniques to speed up the computation can help immensely. With the score-and-search paradigm, one of the most valuable techniques is caching the results of scoring criterion applications. Scoring a structure is normally in $O(nmr)$, where n is the number of variables, m is the number of samples and r is the average number of values per variable. With

caching, this can turn into an operation in $O(n)$. Beyond this very simple, yet effective technique, there lie some other tricks that can help. Below are just a few examples of these.

One of the fundamental operations of learning with the score-and-search paradigm is extracting counts of data from the data set, i.e. finding a contingency table for a certain set of variables. With many variables and a large sample, this can easily become a bottleneck. The *ADtree* data structure described by Moore and Lee (1998) can help in cases where there are a large number of records. It achieves this by not storing zero counts and other redundant information. Another technique by Friedman and Getoor (1999) helps to constrain the number of sufficient statistics (i.e. counts) that need to be collected by using constraints imposed by the statistics already gathered to guide the learning algorithm. The work of Chickering and Heckerman (1999) shows a method to quickly extract one and two way counts from data that can be either real or expected. They also show an algorithm that quickly performs the E step of the EM algorithm. Another technique to speed up EM using a generalised conjugate gradient method is given by Thiesson (1995). Finally, Zhang (1996) shows another modified EM algorithm that works on the principle that some parameters are irrelevant to the probability of seeing a certain datum.

2.4.21 Miscellaneous Techniques

Below are some results by researchers that do not fit neatly into other categories. These are normally ideas that focus on a very particular aspect of the the learning problem.

Local Learning Tsamardinos et al. (2003a,b) focus on learning local features of a Bayesian network. In their case, they focus on direct edges to and from a certain variable and the Markov blanket of a certain variable. These are often very useful structures in the learning of complete Bayesian networks.

Heterogeneous Data Following on from the earlier work of Heckerman (1995a) who looked at learning causal networks, Cooper and Yoo (1999) discuss how to learn structure whilst using a mixture of experimental and observational data.

Operators Moore and Wong (2003) introduce a new operator, optimal reinsertion that allows sufficient statistics to be calculated quickly.

Causal Independence Models Meek and Heckerman (1997) discuss structure and parameter learning of causal independence models, i.e. Bayesian network models where causes of an effect are assumed to be independent from each other. These can be important in modelling situations where the assumption is warranted and also because inference can be cheaper.

Hierarchical Bayesian Networks Gyftodimos and Flach (2004) look at learning hierarchical Bayesian networks, i.e. Bayesian networks, where each node is itself an aggregation of other nodes. Hwang et al. (2006) also mention learning hierarchical Bayesian networks. However their terminology is completely different to Gyftodimos and Flach's. To them, a hierarchical Bayesian network is one where the observed nodes are connected via a hierarchy of hidden nodes. Whilst technically a Bayesian network, it represents knowledge in a very different way than usual. In a sense, there are different *levels* of connectivity between nodes, with higher levels indicating connections between groups of variables.

2.5 Applications

This section aims to look at some typical applications of Bayesian networks. A lot of the original applications were in the medical field and to some extent, this is the domain where Bayesian network applications dominate today. However, there now exist uses in diverse domains, from biology, to natural language processing, to forecasting. Part of the popularity of Bayesian networks must stem from their visual appeal, as it makes them amenable to analysis and modification by experts. However, it is the generality of the formalism that makes them useful across a wide variety of circumstances. Because a Bayesian network is a joint probability distribution, any question that can be posed in a probabilistic form can be answered correctly and with a level of confidence. Some examples of these questions are:

- Given some effects, what were the causes?
- How should something be controlled given current readings?
- What will happen if an intervention is made on a system?

Below are examples of applications across many different domains, that ask in one form or another, questions like these.

Medicine At present, there are many applications of Bayesian networks in medicine. An overview of the field is given by Lucas et al. (2004), but some of the more famous applications are given here.

An early implementation of a system for diagnosis in internal medicine was the quick medical reference (QMR). This system was reformulated in a Bayesian network implementation, with three levels of nodes; background, diseases and symptoms. Known as QMR-DT, it had a very large number of nodes and arcs (Shwe et al., 1991; Middleton et al., 1991). Because of this, algorithms had to be developed that could perform inference in this dense network (Shwe and Cooper, 1991). Another more specific diagnostic system is that from the Pathfinder project (Heckerman et al., 1992), which is used in the diagnosis of lymph-node diseases. In the same vein, but used for diagnosing neuromuscular disorders is the MUNIN network developed by Andreassen et al. (1989).

With a similar domain, but different purpose is the ALARM network developed by Beinlich et al. (1989), which was used for the monitoring of patients in intensive care situations. It is often used as a gold-standard network, as it is reasonably well connected and has enough nodes to be a challenging but still achievable problem for many Bayesian network algorithms. And from a learning perspective, Acid et al. (2004) give a comparison of learning algorithms on the emergency medicine domain.

Forecasting Bayesian networks can be very useful in predicting the future based on current knowledge. One of the most well known of these is the HailFinder network of Abramson et al. (1996), which is used to forecast severe weather. Also in the weather forecasting domain is the sea breeze prediction system of Kennett et al. (2001), which uses learnt structure and probability.

In the market domain, Abramson and Finizza (1991) use a Bayesian network to forecast oil prices, whilst Dagum et al. (1992) show a dynamic Bayesian network used for the same task. And to the extent that classification can be seen as forecasting, Bayesian networks have huge potential. An example of this is by Friedman et al. (1997) who gives a generalisation of the already good naïve-Bayes classifier into the tree-augmented naïve-Bayes classifier. Another implementation of classification using Bayesian networks is by Correa et al. (2007), who use them in the classification stage of an algorithm that also features attribute selection using a discrete particle-swarm optimisation algorithm.

Control An interesting use of dynamic Bayesian networks in the control area is that of Forbes et al. (1995) who showcase their Bayesian automated taxi (BATmobile) network.

This network is in the form of a dynamic influence diagram, and the system as a whole illustrates all the problems that must be solved to provide reliable control in a noisy, partially observed domain.

Modelling for Human Understanding Friedman et al. (2000) and Friedman (2004) look at modelling the causal interactions between genes by analysing gene expression data. They use the sparse candidate algorithm (Friedman et al., 1999c) as described in Section 2.4.8, to learn the structure of 800 genes using 76 samples. These ideas have been built on by Husmeier (2003) who looks at the problem of small sample sizes prevalent with biological data and examines techniques to characterise the sensitivity and specificity of results. Chapter 7 builds on these ideas, by using a dynamic Bayesian network to model the circadian clock of *Arabidopsis Thaliana* using gene expression data.

2.6 Comparison of Techniques and Summary

Given the amount of different techniques that can be used to learn the structure of a Bayesian network, it can be hard to decide which is the best to use in a particular situation. This is not helped by authors failing to give guidelines as to what situations their particular algorithms might be useful in. This section seeks to give a round up of the various techniques discussed in Section 2.4 and the circumstances in which they might be used.

The most important piece of information when deciding what method to use in constructing a Bayesian network structure is probably the use the network will be put to. The main uses of a Bayesian network structure are:

- Provide a DAG that a human can use as a model of the (possibly causal) interactions amongst variables; and
- Coupled with parameter learning, provide a model that can be used to perform inference.

Although methods such as scoring structures and using conditional independence tests can both be used for both of these tasks, there seems to be a slight bias in the literature for using the latter in detecting causal relations. This is because the methods use explicit tests to find these relations; in the case of the scoring methodology, causal semantics are dependent on extra assumptions (Heckerman, 1995a). Also, the CI methods can help in finding hidden variables and selection bias (Spirtes et al., 1995).

With performing inference, there also seems to be a slight bias towards methods that score structures. The reason for this is that they are based on the prequential prediction principle and naturally fall into a good match for the inference task.

2.6.1 Tree and Polytrees

Although they cannot represent the full range of conditional independencies as a DAG can, trees and polytrees might be good enough for a particular task. For example, if there is not enough data to support the high order conditional independencies that can be represented in a graph, then a tree or polytree could be a suitable choice and indeed might not affect the accuracy of the model generated too much (Acid and de Campos, 1995). An advantage of trees is that they can be exactly learnt in polynomial time (Chow and Liu, 1968); polytrees are still NP-hard to learn, but good approximations are easily found (Dasgupta, 1999). Perhaps *the* major advantage of trees and polytrees is that exact inference can be performed in polynomial time (Kim and Pearl, 1983). This can be a large advantage if the time needed for inference to be performed is bounded.

2.6.2 Heuristic Search

Some of the most successful strategies for learning Bayesian networks employ heuristic search whilst scoring network structures. Even simple techniques such as greedy search can produce network structures that are 'good enough'. And as opposed to methods that use conditional independence testing, they can work better with smaller data sets. Perhaps the main reason to use this technique is that it has received the most attention in the literature and hence is more developed.

The main issues when using heuristic search are the search algorithm, the scoring function and the search space. Obviously, greedy algorithms tend to get trapped in local maxima; global search strategies such as genetic algorithms, simulated annealing etc. can produce better solutions at the cost of longer running times.

Deciding on the scoring function to use can be problematic. Bayesian scoring functions such as BDeu produce the best score from the probabilistic sense of anticipating the next datum, but require the specification of priors. On the other hand, large sample approximations such as BIC do not require a prior, but can be slightly inaccurate at small sample sizes. Measures such as the Cheesman-Stutz approximation can help in these situations (Cheesman and Stutz, 1996), and Shaughnessy and Livingston (2005) provide a comparison of different functions.

There are also tradeoffs on deciding on the search space to use. It can be easier and faster to move through the states in the space of DAGs, but there can be plateau effects on the score function, which can make it hard to get to all possible states. Searching through, e.g. the space of equivalence classes of DAGs can avoid this problem, but at the expense of a more complicated implementation and possibly slower running times.

2.6.3 Conditional Independencies

As stated at the start of this section, using conditional independency testing as the basis for structure search is often used when trying to detect causal relations between variables. However, there are problems with small sample sizes, missing data and the fact that a single level of significance must be chosen for the statistical testing of conditional independence. A more interesting use of CI testing may be in mixing it with score and search techniques to produce a hybrid solution to learning structures as mentioned in Section 2.4.8. Here, the testing can be used to massively cut down the search space needing to be searched. This can be very useful when faced with a large number of variables.

2.6.4 Dynamic Programming

A recent addition to the Bayesian network structure-learning toolbox, dynamic programming has enabled feasible exact learning for moderate numbers of variables (up to about 30). With smaller numbers of variables this can be predicted to become the method of choice in applications and a means to generate a standard structure to enable comparisons of new techniques. There also exist techniques to scale above 30 variables, by learning parts of the network in clusters – however in this case, the exactness guarantees do not exist.

Perhaps the main problem with these methods is that they require an exponential amount of space for the memoisation part of the dynamic programming algorithm.

2.6.5 Summary

In this chapter, a broad overview of the literature on learning Bayesian network structures has been given. As a lead up to this, the foundations of Bayesian network theory, along with brief summaries of Bayesian network inference and learning Bayesian network parameters were presented. Also, a look at some applications of Bayesian networks and a

high level comparison of the different methods of learning Bayesian network structures were given.

From the preceding review of the literature on learning Bayesian network structures, some common themes emerge. One of the most pervasive of these themes is the difficulty of learning good structures. Because the problem is NP-hard, heuristic methods generally need to be used, with local methods such as greedy search having the problem of local maxima. In recognition of this, the main body of work of this thesis is focused on avoiding the local optima problem, by using a stochastic method known as ant colony optimisation (ACO). The hope is that in being able to bypass local maxima, better structures can be found that more accurately model the problem domain being examined. Before examining the application of ACO to the problem, a brief look at the background of ACO will be given, in order that a greater appreciation of this technique can be had.

Chapter 3

A Review of the Literature on Ant Colony Optimisation

SINCE the main topic of this thesis is on learning Bayesian network structures using an ant colony optimisation (ACO) algorithm, this chapter will present a brief introduction to the field of ACO. The emphasis of this introduction will be on machine learning applications, in order that a fuller understanding can be had of the motivations and underlying ideas of this relatively new area. This, coupled with Chapter 2, will fully explain the background to the ACO-E algorithm described in Section 4.3, the main contribution of this thesis. To this end, this chapter will be structured as follows.

Firstly, the main motivations of ACO are given, with reference to the study of real-life ants and continuing on to the abstraction to artificial ants. Next, the standard framework for specifying an ACO algorithm is shown, illustrated by a standard example of an ACO algorithm implemented in this framework. Finally, previous work in the area of ACO applied to machine learning tasks is showcased. One of these applications concerns applying ACO to the learning of Bayesian networks. As this application informs the new work in the following chapters, it is looked at in depth.

3.1 Ant Colony Optimisation

Ant colony optimisation is a global optimisation technique generally used in the area of combinatorial problems, i.e. problems where the set of solutions is discrete. Since the inception of its present form by Dorigo (1992), ACO has been successfully applied to many combinatorially hard problems including the sequential ordering problem

(Gambardella and Dorigo, 2000), the vehicle routing problem (Bullnheimer et al., 1999), the bin-packing problem (Levine and Ducatelle, 2004) and many more (Maniezzo and Coloni, 1999; Gambardella and Dorigo, 2000; Stützle, 1998; Costa and Hertz, 1997). Such a diverse range of applications must ask the question as to what is the nature of the system that can solve them.

The particular form of ACO is of a *metaheuristic* in the field of *swarm intelligence* (Bonabeau et al., 1999), that is based on the behaviour of real-life ants as they forage for food. A metaheuristic is a general purpose heuristic that guides other, more problem specific heuristics. Whilst a fuller exposition of metaheuristics will be showcased in Section 3.2, a description of swarm intelligence and the beginnings of ACO as a technique will be given here.

3.1.1 Swarm Intelligence

Swarm intelligence may be defined as:

‘algorithms or distributed problem-solving devices inspired by the collective behaviour of social insect colonies and other animal societies’ (Bonabeau et al., 1999, p. 7)

This statement carries a range of connotations as to how swarm intelligence might be realised. Indeed, there exist methods based on the behaviour of swarms (Kennedy and Eberhart, 1995), foraging behaviour of honey bees (Pham et al., 2006) and the method that will be discussed here, on the foraging behaviour of ants (Dorigo and Stützle, 2004). Because of the broadness of the metaphor, swarm intelligence systems cannot be easily characterised by simple properties. However, some common themes emerge that are shared by multiple realisations of the framework.

One of these themes is *self organisation*. Self organisation is the organisation of a group of agents of some sort, *without a central controller*, i.e. without some agent that directly controls what other agents do. The organisation is in some sense an effect of the agent to agent interactions in the group. This can be seen in the case of particle swarm optimisation (PSO) (Kennedy and Eberhart, 1995), where a flock of agents keeps a coherent structure using rules that are defined in an agent to agent manner.

Another of the themes in swarm intelligence is *stigmergy*. Stigmergy is the indirect communication of agents through the environment and is used in the self organisation of those agents. This effect can be seen in colonies of ants, as they lay down pheromone trails that are followed by other ants, and is the topic of the next section.

3.1.2 Ant Colony Optimisation

Ant colony optimisation is a swarm intelligence technique that is based on the foraging behaviour of real-life ants. In particular, it uses the principle of stigmergy as a communication mechanism. Real-life ants leave a chemical trail behind them as they explore their environment. The substance laid down on this trail is known as *pheromone*. In moving around, ants are more likely to follow a path with more pheromone, than a path with less (or no) pheromone. This behaviour was investigated by Deneubourg et al. (1990), who designed an experiment with a nest of Argentine ants, a food source and two trails between them that could be set to different lengths. Ants would leave the nest, find the food source and return back with food. When the trails were of the same length, it was found that the ants would eventually settle on a single trail for travel to and from the nest. This behaviour can be explained as follows.

When the experiment begins, ants initially choose one of the trails at random. Whilst traversing this trail, they deposit pheromone. This causes following ants to choose the trail the initial ants took more often, and deposit more pheromone on that trail. Again, this causes more following ants to choose the initially chosen trail, *to a greater degree than the first set of following ants*. Put another way, each ant that chooses a certain trail reinforces the probability that following ants will choose that trail. The trail that initially gets chosen by more ants has more pheromone deposited per unit time and hence a positive feedback or *autocatalytic* process is created, where eventually all ants converge to a single trail.

When the trails start out at different lengths, it is found that ants converge on the shorter trail more often than the longer. This can be explained by more ants being able to traverse the shorter trail to the food source and return to the nest in the same amount of time it would take to traverse the longer trail. With more ants traversing the trail, more pheromone is deposited, and the ants eventually converge to that path.

It is the behaviour of the ants when faced with trails of different lengths that ACO is modelled upon. Instead of real-life ants, artificial ants are conceived as a computing unit. Instead of trails, these ants traverse a construction graph. The paths the ants take on this graph are solutions to the problem being looked at – the idea is to reinforce the pheromone on better solutions. However, the fundamental idea of laying down pheromone is kept, with ants depositing it on arcs as they traverse from node to node. Also, ants are programmed to follow arcs with stronger pheromone more often than arcs with weaker pheromone.

Artificial ants can be more useful than real-life ants in that they can be given a memory. This can stop ants looping around and helps when laying pheromone on the return journey. Also they can be programmed to use problem dependent heuristics that can guide the search towards better solutions. All of these ideas and more are discussed in the next part, Section 3.2 and an example of ACO in practice is given in Section 3.3.

3.2 The ACO Metaheuristic

Nowadays, ACO algorithms tend to be defined in terms of the ACO metaheuristic (Dorigo and Di Caro, 1999). A metaheuristic is a general purpose heuristic that guides other, more problem specific heuristics. Examples of metaheuristics include simulated annealing (Kirkpatrick et al., 1983), tabu search (Glover, 1989, 1990), evolutionary computation etc.

In the ACO metaheuristic, a problem is represented by a triple (\mathcal{S}, f, Ω) , where \mathcal{S} is a set of candidate solutions, $f: \mathcal{S} \times T$ is an objective or scoring function that measures a solution's quality at a particular time $t \in T$ and $\Omega: T$ is a set of constraints at time $t \in T$, used in a solution's construction. The range of f and Ω is dependent on the particular instance of the metaheuristic. In trying to map a combinatorial optimisation problem onto this representation, the following framework is used.

- There should be a finite set of solution components $C = \{c_1, c_2, \dots, c_{N_c}\}$. These are the building blocks of candidate solutions.
- The problem states are represented by sequences of solution components $x = \langle c_i, c_j, \dots \rangle$, where $c_i, c_j, \dots \in C$. The set of all possible sequences (states) is given as \mathcal{X} .
- \mathcal{S} – the set of candidate solutions as mentioned above – is a subset of \mathcal{X} , i.e. $\mathcal{S} \subseteq \mathcal{X}$. In this sense, some but not all states are candidate solutions, but all candidate solutions are states.
- There is a set of feasible states $\tilde{\mathcal{X}}$, with $\tilde{\mathcal{X}} \subseteq \mathcal{X}$. A feasible state $x \in \tilde{\mathcal{X}}$ is a state where it is possible to add components from C to x to create a solution satisfying the constraints Ω .
- Each candidate solution $s \in \mathcal{S}$ has a cost $g(s, t)$, where s , being a solution (and hence a state) is a sequence of components, $s = \langle c_i, c_j, \dots \rangle$. Normally $g(s, t) \equiv$

$f(s, t)$, $\forall s \in \tilde{\mathcal{S}}$, where $\tilde{\mathcal{S}} = \mathcal{S} \cap \tilde{\mathcal{X}}$ is the set of feasible candidate solutions. However, this might not always be the case; if f is very expensive to compute, g might be an easier to compute function that is broadly similar to f and that can be used in the generation of solutions.

- The set of optimal solutions \mathcal{S}^* should be non-empty, with $\mathcal{S}^* \subseteq \tilde{\mathcal{S}}$.
- Sometimes it may also be possible to associate a cost $J(x, t)$ to a state $x \in \mathcal{X}$ that is not a candidate solution, that monotonically increases as components are added.

With this framework, solutions to the problem (S, f, Ω) can be generated by having artificial ants perform a *random walk* on the complete graph G defined on the components in C . This graph G is known as the *construction graph*. A random walk on a graph is a series of moves from node to node of the graph, with each move being random to some degree. If the walk is *Markovian*, then the next move is always completely random; if not then the next move is influenced by the previous moves. Hence, using this terminology ACO is non-Markovian. The walk that the ant makes is generally biased by two things – a heuristic value η (η_i if the heuristic is associated with the individual nodes of G , η_{ij} if it is associated with the edges of G) and a pheromone trail τ (again, τ_i if the pheromone is associated with the individual nodes of G , τ_{ij} if the pheromone is associated with the edges of G). The way the heuristic and pheromone are implemented are problem dependent, but in general the heuristic η is a measure of the ‘goodness’ of taking a particular move on the construction graph as defined by some local measure. The pheromone τ is a measure of the ‘goodness’ of taking a particular move as defined by the aggregate behaviour of ants selecting that move and the quality of solutions that these ants generate.

Finally, each artificial ant k has the following properties in order to fully specify how the random walk will proceed:

Memory – Each ant k has a memory \mathcal{M}^k that stores information about the path it has so far followed.

Start State – Each ant k has a start state x_s^k and a non-empty set of termination conditions e^k .

Termination Criteria – When an ant is in a state x , it checks if one of the termination criteria in e^k is satisfied. If not, it moves to a node $j \in \mathcal{N}^k(x)$. \mathcal{N}^k is a function that returns the neighbourhood of a node x , i.e. all the nodes on the construction graph G that can be reached from the current state, given the constraints Ω .

Decision Rule – An ant chooses to move to the node j according to a probabilistic decision rule, which is a function of the pheromone τ and the heuristic η . The specification of these rules is problem dependent, but is usually a random choice biased towards moves with a higher heuristic and pheromone.

Pheromone Update – The pheromone of a path can be modified by an ant as it is traversing it, or on the return journey, when it returns to the start. Again, this is problem dependent, but a standard formulation is to increase the pheromone on good solutions and decrease the pheromone on bad solutions, good and bad being given by the specific formulation.

In terms of algorithmic actions, an ACO algorithm can normally be broken down into three parts. These are:

ConstructAntsSolutions This part of the algorithm is concerned with sending ants around the construction graph according to the rules given above.

UpdatePheromones This part is concerned with changing the values of the pheromones, by both depositing and evaporating. Parts of this task might be performed during an ant's traversal of the graph, when an ant's traversal is finished or after an iteration of all the ants' traversals.

DaemonActions This part of the algorithm performs tasks not directly related to the ants. E.g. a local search procedure might be performed after each ant finishes its traversal.

3.3 Example – The Travelling Salesman Problem

Given the above framework, multiple artificial ants are released to perform a random walk. This procedure is repeated a number of times, with the pheromone gradually increasing on the best parts of the solution. To illustrate the metaheuristic, a concrete problem will now be presented with reference to the various parts of the framework introduced above.

Perhaps because of its physical resemblance to ants walking around, the travelling salesman problem (TSP) was one of the first to be studied using an algorithm called Ant System (Dorigo et al., 1996). In this problem, a number of cities and costs of travelling between cities are given, with the problem being to find the cheapest tour that visits all

the cities exactly once and then returns to the start city. The TSP was used to introduce most of the fundamental ideas used in ACO today and hence is often used in illustrating new ACO techniques.

According to the framework given above the problem is given as (\mathcal{S}, f, Ω) where:

- \mathcal{S} is the set of all possible Hamiltonian circuits of the construction graph;
- f is the cost of a solution $s \in \mathcal{S}$, i.e. the sum of all trip distances on the tour; and
- Ω is the constraint that ants only construct paths that contain each city once.

With the problem defined, the solution components can be given as follows:

- C is the set of all cities, which gives the construction graph $G = (C, L)$ where L are the weighted arcs on the complete graph, the weights being the cost of going between each pair of cities;
- \mathcal{X} , the set of all states, is given by all sequences of the components C ;
- $\tilde{\mathcal{X}}$, the set of all feasible states, is given as all sequences of length less than or equal to $n + 1$ where n is the number of cities and no value in the sequence is repeated, except that $x_1 = x_{n+1}$;
- \mathcal{S}^* is the set of Hamiltonian cycles that have the smallest cost;
- $g(s, t) \equiv f(s, t)$; and
- $J(x, t) = \sum_{i=1}^{|x|-1} d_{x_i x_{(i+1)}}$, i.e. the sum of the distances between adjacent cities in the sequence.

Each of the ants that construct a solution is very simple, with ant k having:

- a memory \mathcal{M}^k that is simply a state $x \in \tilde{\mathcal{X}}$;
- a start state $x_s^k = \langle \rangle$;
- a termination condition e^k which is true if $|x| = n + 1$; and
- a neighbourhood $\mathcal{N}^k(x)$ which is $C \setminus \{x\}$, except when $|x| = n$ where it is $\{x_1\}$.

Each ant also has a probabilistic decision rule,

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} \quad (3.1)$$

This says that the probability of taking a particular path is proportional to the pheromone on that path and heuristic desirability of that path. If ant k is at node i , then p_{ij}^k is the probability that it will traverse the arc connecting i to j . Here, τ_{ij} and η_{ij} are the pheromone and heuristic associated with the arc between i and j and α and β are user defined parameters that specify the relative importance of the pheromone and heuristic in selecting a path to follow. The heuristic η_{ij} is defined as $\eta_{ij} = 1/d_{ij}$, where d_{ij} is a distance measure between two nodes i and j . The numerator in Equation 3.1 gives the weight of a particular path, whilst the denominator is the sum of the weightings of all the paths and ensures that p_{ij}^k can be interpreted as a probability.

After all the ants have constructed their tour, pheromone is then evaporated from all the arcs on the construction graph G ,

$$\tau_{ij} \leftarrow (1 - \rho) \tau_{ij}, \quad \forall (i, j) \in L,$$

with $\rho \in [0, 1]$ being the proportion of pheromone that is evaporated. Finally, all ants deposit pheromone on the path they traversed as below

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^m \Delta \tau_{ij}^k, \quad \forall (i, j) \in L,$$

where m is the number of ants traversing the construction graph and $\Delta \tau_{ij}^k$ is the amount of pheromone ant k deposits on the arc (i, j) and is given by

$$\Delta \tau_{ij}^k = \begin{cases} 1/C^k, & \text{if arc } (i, j) \text{ belongs to } T^k \\ 0, & \text{otherwise} \end{cases},$$

where C^k is the cost or length of the tour T^k made by ant k . The above system of constructing tours is repeated a certain number of iterations, with this number being constant, or until no progress has been made in the last h iterations.

Advances in ACO generally use the framework described above. One of the newer systems which has improved on Ant System is the $\mathcal{MAX}-\mathcal{MIN}$ Ant System (\mathcal{MMAS}) (Stützle and Hoos, 2000). In this system, only one ant deposits pheromone, either the iteration-best ant or the best-so-far ant. The iteration-best ant is the ant that constructed the best solution in the current iteration; the best-so-far ant is the ant that has currently

constructed the best solution up to this point in the algorithm run. Secondly there are limits on the values the pheromone can take in order to guard against stagnation. This is the problem that occurs when pheromone levels on certain trails are very high in relation to others and effectively stop other solutions being generated. Thirdly, the pheromone trails are initialised at a high value and reinitialised after no improved tour has been generated for a number of iterations. Another extension to Ant System is the Ant Colony System (ACS) (Dorigo and Gambardella, 1997). Here, the search is more biased towards the best-so-far path, with a pseudo-random proportional decision rule that takes the best solution component most of the time and the normal random proportional decision rule the rest of the time. Also, like *MMAS* only the best-so-far ant deposits pheromone. ACS is based on a system known as ANT-Q designed by Gambardella and Dorigo (1995), that is itself inspired by the reinforcement learning technique of Q-learning (Sutton and Barto, 1998). The ACS is particularly interesting in this context, as it is the system on which the new work described in later sections has been modelled. This is because this work is inspired by a previous approach to learning Bayesian networks using ACO (described in Section 3.5) which used ACS as its form of ACO.

3.4 ACO in Machine Learning Problems

Although the main area that ACO tends to be used in is combinatorial optimisation problems, it has also been used with success in the area of problems that might traditionally be called machine learning. One feature that machine learning problems sometimes have is that solutions are generated iteratively, i.e. there is a move from state to state in a solution space. This carries over to ACO, with the effect that each solution state *is* a candidate solution. In this situation there needs to be some stopping criterion, but this is normally handled by there being no improvement according to the objective function.

In general, from a machine learning perspective, optimisation is often seen as a sub-field, with the broad theme of machine learning being the ability to model a system with good generalisation capability; optimisation is often used to generate good models to a problem that has been constructed (Bennett and Parrado-Hernández, 2006).

Neural Networks Application of ACO to machine learning type problems is relatively new in the field, but in the last few years a diverse range has appeared. One example of this is in learning the weights of a feed-forward neural network. The standard technique for this is the back-propagation algorithm, but this has problems with local minima. ACO,

being a global learning technique, has been applied to this problem by Blum and Socha (2005). Because the weights of the links between neurons are real numbers, it might seem hard to fit this into the learning scheme of ACO which deals with combinatorial optimisation. However, they describe a method where instead of an ant choosing a neighbouring node in a construction graph parameterised by a pheromone matrix, the ant chooses a real number depending on a Gaussian kernel probability density. This density is constructed from a population of the solutions kept by the algorithm.

Another work in the learning of feed-forward neural network weights is by Liu et al. (2006). They take a more conventional view of the problem, by discretising the possible weights and using these as values of the variables in an ACS type system. A given solution can then be used as a starting point for a standard back-propagation algorithm.

Feature Selection With data sets containing many attributes, there is often a need to reduce the amount of attributes, both because of the problem of noisy redundant attributes making algorithms perform poorly and to speed up processing of further algorithms. A standard approach to this problem is principal component analysis (PCA), that returns a modified subspace of the original space. This idea is expanded upon by Yan and Yuan (2004) who use feature selection in the task of face recognition. They firstly perform PCA on the data, extract the 100 features with the highest eigenvalues and then use an ACO algorithm to select the best subset of features.

Clustering Another standard machine learning task is clustering, i.e. assigning a label to each datum in a data set, such that data with the same label are 'close together'. A standard approach to this task is the k -means algorithm; this however often produces a local optimum. Runkler (2005) demonstrates an approach to the hard c-means and fuzzy c-means tasks that uses ACO.

Regression Whilst clustering is considered an *unsupervised* learning task, there also exist examples of ACO being used in *supervised* learning tasks. An example of this is the work of Hong et al. (2007) who demonstrate a system that uses ACO to learn the parameters of a support vector regression (SVR) model; i.e. the use of support vector machines to solve non-linear regression problems.

Classification A much earlier body of work on the subject is that by Parpinelli, Lopes and Freitas (2001, 2002a,b,c). In this, the authors describe an algorithm known as

AntMiner that uses ACO to build a classifier system consisting of an ordered set of if-then rules. In this case, the construction graph is complete over all possible terms (attribute instantiations), with the pheromone associated with the nodes, rather than the edges. Rules are constructed by adding terms to a current rule, dependent on the pheromone and heuristic and then pruning rules before the class term predicted by the rule is set. The best rule in each iteration is added to the rule set, until the number of uncovered cases in the training data falls below a given parameter. Although the algorithm is conceptually quite simple, it performs well compared to CN2 (Clark and Niblett, 1989) and C4.5 (Quinlan, 1993), especially in the length of rules it produces.

The results of Parpinelli, Lopes and Freitas have been built upon by other authors to produce varying types of systems. Chan and Freitas (2006b) describe a system that produces rules that contain multiple class attributes. The work of Smaldon and Freitas (2006) is focused on building unordered rule sets, as opposed to the ordered rule sets of the original work. Chan and Freitas (2006a) discuss a new pruning procedure for the AntMiner system that produces shorter but slightly less accurate rule sets. Holden and Freitas (2004) show an application of AntMiner to web-content mining.

Continuous Features Although attempts have been made to extend ACO to non-combinatorial optimisation problems (i.e. optimisation problems that are parameterised by continuous values and where the solution space is continuous) (Blum and Socha, 2005; Hong et al., 2007), work in this area is quite new. This can be a problem when dealing with real-valued attributes, especially in a system such as AntMiner. Although discretisation can be used, the discretisation method and parameters can cause results to widely vary, and it can be hard to know which results to trust. Another solution to the problem is to mix methods in an algorithm and have one to deal with continuous attributes and one to deal with categorical. This is the approach followed by Holden and Freitas (2005, 2006, 2007), who create a hybrid PSO/ACO algorithm in a manner similar to AntMiner.

3.5 ACO in Learning Bayesian Network Structures

Whilst ACO has been applied to many problems in the area of combinatorial optimisation, to date there has not been much research on using the technique to learn Bayesian network structures. Two alternate methods have been defined by de Campos et al. (2002a,b). The first conducts a search in the space of orderings of DAGs, whilst the second searches

in the space of DAGs. Both of these spaces have been discussed in Sections 2.4.4 and 2.4.9. Since a main topic in this thesis is on this problem, a description of both of these will be given here, in order to examine the early work done on the subject and see how it can inform future studies.

3.5.1 ACO-K2SN

In the first technique, known as ACO-K2SN, searching over the space of orderings of DAGs, the various problem components, as taken from Section 3.2, can be defined as follows:

Construction Graph There is one node for each attribute in the data, with an extra dummy node from which the search starts.

Constraints The only constraints are that the tour is a Hamiltonian path, i.e. a path that visits each node exactly once.

Pheromone Trails The pheromone is associated with each arc on the graph. Each arc in the graph is initialised to a initial small value.

Heuristic Information The heuristic on each arc is set to the inverse of the K2 score that is explained below.

Solution Construction The ants work on a system very similar to the ACS system. Beginning at the dummy node, the ants construct a complete path that defines an ordering of the nodes.

Pheromone Update This works exactly as in ACS, with local pheromone updates and global update on the best so far solution.

Local Search A version of local search on orderings known as HCSN (de Campos and Puerta, 2001a). This is used on the last iteration of the run.

Given the above components, the search for an ordering proceeds as follows. Starting at the dummy node an ant decides which node to go to next. This will be the first node in the ordering. To choose a node, heuristic information and pheromone is used. The heuristic for the arc from i to j is given by

$$\eta_{ij} = \frac{1}{|f(x_j, Pa(x_j))|},$$

where f is the K2 scoring metric as explained in Section 4.1 and $Pa(x_j)$, the parents of x_j are found by the K2 algorithm, with possible parents being the nodes already visited. The initial pheromone value τ_0 is given by

$$\tau_0 = \frac{1}{n|f(S_{K2SN})|},$$

where S_{K2SN} is the structure given by the K2SN algorithm of de Campos and Puerta (2001b). The update value for the pheromone is given by

$$\Delta\tau_{ij} = \frac{1}{|f(S^+)|},$$

where S^+ is the best-so-far structure.

3.5.2 ACO-B

The second algorithm given by de Campos et al. is the ACO-B algorithm. The components for this algorithm are:

Construction Graph There is one node for each possible directed arc between each pair of attributes (excluding self directed arcs). There is also a dummy node that the ants start from.

Constraints The only constraints are that the DAG must be acyclic at each step.

Pheromone Trails The pheromone is associated with each node on the graph. The pheromone at node (i, j) corresponds to the directed arc $j \rightarrow i$.

Heuristic Information The heuristic on each node (i, j) is the gain in score that would occur in adding an arc $j \rightarrow i$.

Solution Construction The ants work on a system very similar to the ACS system. Beginning at the dummy node, the ants construct a path that defines which arcs are added to the DAG. This process ends when there is no gain in score.

Pheromone Update This works exactly as ACS, with local pheromone updates and global update on the best so far solution.

Local Search A standard greedy search with arc addition, deletion and reversal is carried out on the current candidate DAG. This is done every 10 iterations.

As opposed to the ACO-K2SN algorithm given in Section 3.5.1, the search is over the space of DAGs, not orderings of nodes. Otherwise, there are some similarities in the definitions of parts of the algorithm. The heuristic is given by

$$\eta_{ij} = f(x_i, Pa(x_i) \cup \{x_j\}) - f(x_i, Pa(x_j)),$$

that is, the change in score by adding an arc from j to i in the candidate DAG. The initial pheromone is given by

$$\tau_0 = \frac{1}{n |f(S_{K2SN})|},$$

i.e. it is the same as the heuristic in ACO-K2SN. Also, the pheromone update value is the same as in ACO-K2SN i.e.

$$\Delta\tau_{ij} = \frac{1}{|f(S^+)|}$$

3.5.3 Performance Comparison

In the results given in both (de Campos et al., 2002b) and (de Campos et al., 2002a), the ACO-B algorithm performs slightly better in terms of accuracy than ACO-K2SN across the ALARM (Beinlich et al., 1989) and INSURANCE (van der Putten and van Someren, 2004) gold-standard networks. It also contains an order of magnitude fewer statistical tests and so should always be faster. There are more comparisons of ACO-B against other algorithms in (de Campos et al., 2002a). Here, it is compared against ILS, an iterative local search algorithm with random perturbations of a local maximum and two estimation of distribution (EDA) genetic algorithms, the univariate marginal distribution algorithm (UMDA) by Mühlenbein (1997) and the population-based incremental learning algorithm (PBIL) by Baluja (1994). Compared across the ALARM, INSURANCE and BOBLO (Rasmussen, 1995) networks, ACO-B performed better than the other methods.

3.6 Summary

This chapter discussed the main features of the ant colony optimisation (ACO) meta-heuristic in order to give a background into this technique, that will be used as a basis of the main work in this thesis.

To begin with, an overview was given of ACO and how it relates to the more general framework of swarm intelligence. The behaviour of ants foraging for food was given as the inspiration for the technique. The jump from real to imaginary ants was then made.

A modern method of defining ACO as a metaheuristic was then introduced, along with a particular instantiation of this metaheuristic involving the travelling salesman problem.

The chapter then focused on ACO being applied to machine learning type problems. In particular the application of ACO to learning the structure of Bayesian networks was looked at, as this is one of the main topics of this thesis. Two methods were considered, which conducted searches in the space of DAGs and orderings of DAGs.

As will be discussed in Section 4.1.2, there are disadvantages to conducting searches in the space of DAGs. To this end, the next chapters will discuss a new technique that can search in the space of equivalence classes of DAGs. This technique is inspired by the approaches looked at in this chapter to learning Bayesian network structures by ACO. The method will be tested and it will be shown that the performance is better than the approaches in this chapter and indeed, many other state-of-the-art approaches in the literature.

Chapter 4

Using ACO in the Learning of Bayesian Network Equivalence Classes

IN THIS CHAPTER and the next two chapters, the main body of work on learning Bayesian networks using ant colony optimisation will be presented. This chapter will deal with the theory behind the work. Chapter 5 will introduce the experiments conducted using this theory, the methodology of these experiments and the results obtained. Chapter 6 will give a discussion and interpretation of the results.

The work described in these chapters is divided into two strands. The first strand deals with methods to speed up the learning of Bayesian network structures by a search through the space of equivalence classes. These methods were originally devised to speed up the running of experiments of the ACO algorithm described in this chapter (ACO-E). However, they stand by themselves as a generic technique and for this reason will be described separately. As these methods are used as a subroutine, they will be discussed first.

The second strand uses ACO in learning Bayesian network structures by searching through the space of equivalence classes. This work is based on that of Chickering (2002a) in describing the space of equivalence classes of Bayesian network structures that can be efficiently used to perform a search procedure in order to learn a Bayesian network structure. It is also related to the ACO-B algorithm of de Campos et al. (2002a), that uses ACO in the space of DAGs to learn a Bayesian network structure. Bringing these two techniques together allows a global search that performs better in terms of network

reconstruction than many other state-of-the-art algorithms.

This chapter will begin with a more in-depth look at the background on learning Bayesian network structures than was given in earlier chapters. This will focus on learning in the space of equivalence classes. The concepts introduced will later be used when discussing the two strands of work mentioned above. After introducing the background, the theory on accelerating the learning process will be given. Finally the theory on using ACO in learning Bayesian network structures will be presented.

4.1 Score and Search Based Methods of Learning Bayesian Network Structures

In learning a Bayesian network from data, both the structure \mathcal{G} and parameters Θ must be learned. These must normally be done separately. In the case of complete multinomial data, the problem of learning the parameters is easy, with a simple closed form formula for Θ (Heckerman, 1995b). However, in the case of learning the structure, no such formula exists and other methods are needed. In fact, learning the structure is an NP-Hard problem and consequently enumeration and test of all network structures is not likely to succeed (Chickering, 1996a). With just ten variables there are roughly 10^{18} possible DAGs. Whilst there exist dynamic programming methods that can handle roughly 30 variables as discussed in Section 2.4.10, in general, non-exact methods are possibly the only tractable solution to anything above this.

In order to create a space in which to search through, three components are needed:

- Firstly all the possible solutions must be identified as the set of states in the space.
- Secondly a representation mechanism for each state is needed.
- Finally a set of operators must be given, in order to move from state to state in the space.

Once the search space has been defined, two other pieces are needed to complete the search algorithm:

- A scoring function which evaluates the ‘goodness of fit’ of a structure with a set of data.
- A search procedure that decides which operator to apply, normally using the scoring function to see how good a particular operator application might be.

An example of a search procedure is greedy search, that at every stage applies the operator that produces the best change in the structure, according to the scoring function. As for the scoring function, various formulæ have been found to see how well a DAG fits a data sample.

One of these functions is given by computing the posterior probability of a structure \mathcal{G} given a sample of data D , i.e.

$$S(\mathcal{G}, D) = P(\mathcal{G}|D) = \frac{P(D|\mathcal{G})P(\mathcal{G})}{P(D)}. \quad (4.1)$$

Here the value $P(D)$ is a constant across all network structures and so can be ignored. This gives $S(\mathcal{G}, D) = P(\mathcal{G}, D) = P(D|\mathcal{G})P(\mathcal{G})$, i.e. the relative posterior probability.

The likelihood term above can take many forms. One popular method is called the Bayesian Dirichlet (BD) metric. Here,

$$P(D|\mathcal{G}) = \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})} \quad (4.2)$$

In this formula, there are n variables in the graph, so the first product is over each variable. There are q_i configurations of the parents of node i , so the second product is over all possible parent configurations, i.e. the cross product of the number of possible values each parent variable can take on. Each variable i can take on one of r_i possible values. The value N_{ijk} is the number of times that variable $i = k$ and the parents of i are in configuration j in the data sample D . N_{ij} is given as $\sum_{k=1}^{r_i} N_{ijk}$, i.e. the sum of N_{ijk} over all possible values that i can take on. With $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$, the values N'_{ijk} are given as parameters that give different variants of the BD metric. E.g. if N'_{ijk} is set to 1 the K2 metric results, as given by Cooper and Herskovits (1992). With N'_{ijk} set to $N'/(r_i \cdot q_i)$ (where N' , known as the equivalent sample size, is a measure of the confidence in the prior value $P(\mathcal{G})$ (Heckerman et al., 1995)), the BDeu metric results which was proposed by Buntine (1991) and further generalised by Heckerman et al. (1995).

The prior value $P(\mathcal{G})$ is a measure of how probable a particular structure is before any data is seen. These values can often be hard to estimate because of the massive numbers of graphs, each of them needing a probability. Therefore, the values are often given as uniform over all possible network structures, possibly favouring structures with fewer arcs.

Other forms used for the scoring function are $S(\mathcal{G}, D) = \log P(D|\mathcal{G}, \hat{\Theta}) - \frac{d}{2} \log N$, known as the Bayesian information criterion (BIC) (Schwarz, 1978) and $S(\mathcal{G}, D) = \log P(D|\mathcal{G}, \hat{\Theta}) - d$, known as the Akaike Information Criterion (AIC) (Akaike, 1974).













Operator	Before		After	
Insert_Arc(X,Y)				
Delete_Arc(X,Y)				
Reverse_Arc(X,Y)				

Table 4.1: Basic modification operators

In these models, the parameters $\hat{\Theta}$ give the maximum likelihood estimate of the likelihood, d is the number of free parameters in the structure and N is the number of samples in the data D .

Traditionally, in searching for a Bayesian network structure, the set of states is the set of possible Bayesian network structures, the representation is a DAG and the set of operators are various small local changes to a DAG, e.g. adding, removing or reversing an arc, as illustrated in Table 4.1. This type of search is possible because of the decomposition properties of score functions,

$$S(\mathcal{G}, D) = \prod_{i=1}^n s(v_i, \text{Pa}^{\mathcal{G}}(v_i), D),$$

where s is a scoring function that takes a node v_i and the parents of this node in graph \mathcal{G} , $\text{Pa}^{\mathcal{G}}(v_i)$. Popular scoring functions such as the BD metric are decomposable in this manner. If

$$S(\mathcal{G}, D) = P(D|\mathcal{G})P(\mathcal{G}),$$

and since the logarithm is a monotonically increasing function, the scoring function S can be redefined to

$$\begin{aligned} S(\mathcal{G}, D) &= \log(P(D|\mathcal{G})P(\mathcal{G})) \\ &= \log P(D|\mathcal{G}) + \log P(\mathcal{G}). \end{aligned}$$

Now by the likelihood given in equation 4.2,

$$\log P(D|\mathcal{G}) = \sum_{i=1}^n \log \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})}$$

Therefore, for the BD metric,

$$s(v_i, \text{Pa}^G(v_i), D) = \log \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})}$$

The log function is often distributed into the right hand term of this equation in order to avoid the instability of the gamma function at high values, giving

$$s(v_i, \text{Pa}^G(v_i), D) = \sum_{j=1}^{q_i} \log \Gamma(N'_{ij}) - \log \Gamma(N'_{ij} + N_{ij}) + \sum_{k=1}^{r_i} \log \Gamma(N'_{ijk} + N_{ijk}) - \log \Gamma(N'_{ijk})$$

Successful application of the operators is also dependent on the changed graph being a DAG, i.e. that no cycle is formed in applying the operator.

4.1.1 Techniques for Searching Through Equivalence Classes

Note that below, a *move* is referred to as an application of an operator to a particular state in the search space.

To be able to conduct a search through the space of equivalence classes (as defined in Section 2.1.3), a method must be able to find out whether a particular move is valid and if valid, how good that move is. These tasks are relatively easy whilst searching through the space of DAGs – a check whether a move is valid is equivalent to a check whether a move keeps a DAG acyclic. The goodness of such a move is found out by using the scoring function, but rather than scoring each neighbouring DAG in the search space, the decomposability of most scoring criteria can be taken advantage of, with the result that only nodes whose parent sets have changed need to be scored.

However, this task of checking move validity and move score is not as easy in the space of equivalence classes. These classes are often represented by PDAGs, as discussed in Section 2.1.3. For one, instead of just checking for cycles, checks also have to be made so that unintended v-structures are not created in a consistent extension of a PDAG. Scoring a move also creates difficulties, as it is hard to know what extension and hence what changes in parent sets of nodes will occur, without actually performing this extension. Also, a local change in a PDAG *might* make a non-local change in a corresponding consistent extension and so force unnecessary applications of the score function.

These problems were voiced as concerns by Chickering (1996b). In that paper, validity checking of moves is performed by trying to obtain a consistent extension of the resulting PDAG – if none exists then the move is not valid. Scoring the move was achieved by scoring the changed nodes in the consistent extension given. These methods were very

generic, but resulted in a significant slowdown in algorithm execution, compared to search in the space of DAGs.

To alleviate this problem, authors proposed improvements that would allow move validity and move score to be computed without needing to obtain a consistent extension of the PDAG (Munteanu and Bendou, 2001; Chickering, 2002a). This was done by defining an explicit set of operators, with each operator having a validity test and corresponding score change function, that could be calculated on the PDAG. These changes led to a speed up of the execution time of the algorithm, with the result that search in the space of equivalence classes of Bayesian networks became competitive with search in the space of Bayesian networks. An example of one set of these operators is given in Table 4.2. In this table, the variables x and y refer to nodes in a graph. As an example, the InsertU operator takes two nodes as arguments, x and y . It can be seen that all the operators take two arguments, except MakeV, which takes three arguments. Each operator also has a set of validity tests that must be passed in order for the application of the operator with its particular arguments to be valid. Finally, the score difference between the old and new PDAGs is given in the last column. In this table:

Π_x is the parent set of node x , i.e. the set of nodes that have directed arcs going to node x ;

N_x is the neighbour set of node x , i.e. the set of nodes that have undirected arcs going to node x ;

$N_{x,y}$ is the set of shared neighbours of nodes x and y , i.e. $N_x \cap N_y$; and

$\Omega_{x,y}$ is the set of parents of x that are neighbours of y , i.e. $\Pi_x \cap N_y$.

Also, as a convenience,

M^{+x} is notation for $M \cup \{x\}$; and

M^{-x} is notation for $M \setminus \{x\}$.

This notation and the set of operators in Table 4.2 come from those proposed by Chickering (2002a). Other definitions include:

- An undirected path is a path from one node to another that only follows undirected edges;

- A semi-directed path is a path from one node to another that only follows undirected edges or directed edges from tail to head; and
- A set of nodes N is a clique, if it is a completely connected subgraph of a graph, (i.e. every node is connected to every other).

4.1.2 Advantages of Searching in E-space

With this representation of equivalence classes of Bayesian network structures and a set of operators that modify the CPDAGs which represent them (e.g. insert an undirected arc, insert a directed arc etc.), a search procedure can proceed. However, what reasons are there for pursuing this type of search? Chickering (2002a) gives a list of reasons, some of which are discussed here.

For one, an equivalence class can represent many different DAGs in a single structure. With a DAG representation, time can be wasted rescoring DAGs that are in the same equivalence class. And with a search in the space of DAGs, the connectivity of the search space can mean that the ability to move to a particular neighbouring equivalence class can be constrained by the particular representation given by a DAG. There is also the problem given by the prior probability used in the scoring function. Whilst searching through the space of DAGs, certain equivalence classes can be over represented by this prior, because there are many more DAGs contained in the class. An example can be given in the case of networks with two nodes. In B-space there are 3 possible structures, which with equal priors give $P(\mathcal{G}) = 1/3$, for each DAG \mathcal{G} . However, the two DAGs that are connected represent the same equivalence class, giving it an effective prior of $2/3$. In E-space there are 2 possible structures, which with equal priors give $P(\mathcal{P}) = 1/2$, for each PDAG \mathcal{P} .

These concerns have motivated researchers. In particular, recent implementations of algorithms that search through the space of equivalence classes have produced results that show a marked improvement in execution time and a small improvement in learning accuracy, depending on the type of data set (Chickering, 2002a,b). These ideas will be used in the work to be discussed later in this chapter.

4.2 Accelerating the Learning Process

Whilst the execution time of searching for equivalence classes of Bayesian networks has decreased, it still remains quite high for problem instances with many variables. This is

especially so if the search algorithm needs multiple traversals through the search space. Therefore, a method of speeding up performance is to cache the values of the score function – this normally gives a large cut in execution time. Since this technique is so easily implemented, it is standard fare for score-and-search algorithms.

With this caching enabled, multiple identical runs over a typical greedy search using the operators given by Chickering (in Table 4.2) were analysed. From this analysis, it was found that much of the time was spent computing two main quantities: score values and validity tests.

In the first run, the dominant factor was the time needed to compute the values given by the score function. This was typically in the order of 90% of the running time. However, in the succeeding runs, it was found that most of the execution time was used in calculating the validity tests for the various operators. The situation had become inverted so that this checking was taking roughly 90% of the time. This occurred because values given by the score function had been cached. In particular, checking the validity conditions for the operators InsertU, InsertD and MakeV was taking the most time.

Upon further analysis, it was seen that checking the ‘path’ condition in each of these operators was the main culprit. Each of these path conditions involves a depth or breadth-first search of the graph. Normally, this is a fairly quick operation, but the test needed to be repeated for each pair of nodes, meaning there were in the order of n^2 searches for each operator. In order to reduce this time taken, two new methods were examined. These methods follow on from those introduced by Daly et al. (2006a) and Daly and Shen (2007).

The first of these methods seeks to reduce the amount of checks performed. It does this by reformulating the validity tests of the operators so they are based on nodes rather than pairs of nodes. The second method is a system that stores and removes valid moves from a cache in order to avoid repeating tests.

4.2.1 Reducing the Number of Checked Nodes

As shown in Table 4.2, five of the operators take two nodes as parameters and one takes three nodes. In the naïve case, where $n = |V|$ (the number of nodes in the graph), this would mean $O(n^2)$ checks. If the validity test includes a path condition, this will take time in $O(n + e)$, where e is the number of edges on the graph. This could mean time in $O(n^3 + n^2e)$ to check the operators.

However, looking at the validity tests more closely, it can be seen that not all combi-

Operator	Effect	Validity Tests	Change in Score
InsertU $x - y$	Add an undirected arc between x and y	<ol style="list-style-type: none"> Every undirected path from x to y contains a node in $N_{x,y}$ $\Pi_x = \Pi_y$ 	$s(y, N_{x,y}^{+x} \cup \Pi_y)$ $-s(y, N_{x,y} \cup \Pi_y)$
DeleteU $x - y$	Delete an undirected arc between x and y	$N_{x,y}$ is a clique	$s(y, N_{x,y} \cup \Pi_y)$ $-s(y, N_{x,y}^{+x} \cup \Pi_y)$
InsertD $x \rightarrow y$	Add a directed arc from x to y	<ol style="list-style-type: none"> Every semi-directed path from y to x contains a node in $\Omega_{x,y}$ $\Omega_{x,y}$ is a clique $\Pi_x \neq \Pi_y$ 	$s(y, \Omega_{x,y} \cup \Pi_y^{+x})$ $-s(y, \Omega_{x,y} \cup \Pi_y)$
DeleteD $x \rightarrow y$	Delete a directed arc from x to y	N_y is a clique	$s(y, N_y \cup \Pi_y^{+x})$ $-s(y, N_y \cup \Pi_y)$
ReverseD $x \rightarrow y$	Reverse a directed arc from x to y	<ol style="list-style-type: none"> Every semi-directed path from x to y that does not include the edge $x \rightarrow y$ contains a node in $\Omega_{y,x} \cup N_y$ $\Omega_{y,x}$ is a clique 	$s(y, \Pi_y^{+x})$ $+s(x, \Pi_x^{+y} \cup \Omega_{y,x})$ $-s(y, \Pi_y)$ $-s(x, \Pi_x \cup \Omega_{y,x})$
MakeV $x \rightarrow z \leftarrow y$	Direct undirected arcs from x and y to z	Every undirected path between x and y contains a node in $N_{x,y}$	$s(z, \Pi_z^{+y} \cup N_{x,y}^{-z+x})$ $+s(y, \Pi_y \cup N_{x,y}^{-z})$ $-s(z, \Pi_z \cup N_{x,y}^{-z+x})$ $-s(y, \Pi_y \cup N_{x,y})$

Table 4.2: Validity conditions and change in score for each operator

Operator	Validity Tests	VALID-NODES
InsertU x	<ol style="list-style-type: none"> Every undirected path from x to y contains a node in $N_{x,y}$ $\Pi_x = \Pi_y$ 	<ol style="list-style-type: none"> $V \setminus X_N \cup \text{CHECK}(x, N_{N_x} \setminus (N_x \cup \{x\}))$ $\begin{cases} \{\xi \xi \in V, \xi \neq x, \Pi_\xi = 0\} & \text{if } \Pi_x = 0 \\ \text{CHECK}(x, \Xi_{\Pi_x}) & \text{otherwise} \end{cases}$
DeleteU x	$N_{x,y}$ is a clique	$\text{CHECK}(x, N_x)$
InsertD x	<ol style="list-style-type: none"> Every semi-directed path from y to x contains a node in $\Omega_{x,y}$ $\Omega_{x,y}$ is a clique $\Pi_x \neq \Pi_y$ 	<ol style="list-style-type: none"> $V \setminus X_{N\Pi} \cup \text{CHECK}(x, N_{\Pi_x} \setminus \Pi_x)$ $V \setminus X_{N\Pi} \cup \text{CHECK}(x, N_{\Pi_x} \setminus \Pi_x)$ $\begin{cases} \{\xi \xi \in V, \Pi_\xi \neq 0\} & \text{if } \Pi_x = 0 \\ \text{CHECK}(x, V \setminus \Pi_x \cup \{x\}) & \text{otherwise} \end{cases}$
DeleteD x	N_y is a clique	$\text{CHECK}(\Xi_x)$
ReverseD x	<ol style="list-style-type: none"> Every semi-directed path from x to y that does not include the edge $x \rightarrow y$ contains a node in $\Omega_{y,x} \cup N_y$ $\Omega_{y,x}$ is a clique 	<ol style="list-style-type: none"> $\text{CHECK}(x, \Xi_x)$ $\text{CHECK}(x, \Xi_x)$
MakeV x	Every undirected path between x and y contains a node in $N_{x,y}$	$\text{CHECK}(x, N_{N_x} \setminus (N_x \cup \{x\}))$

Table 4.3: Validity conditions and set of valid nodes for a node x

nations of nodes need be checked. In particular, given a node x , two subsets of the nodes V can be found

- A subset for which a move is valid; and
- A subset that needs to be checked by the original conditions.

E.g. with the InsertU operator, it is obvious if a node y is not in the same connected component as a node x then there can be no undirected path between the nodes and hence the first validity test is fulfilled.

Table 4.3 shows for each operator, the original validity tests used, and the set of nodes for which this test is valid. The first column of the table shows the various operators as defined in Table 4.2, except that instead of pairs or triples, they each take a single argument x . The second column shows the various validity tests for which an instance of a move would be valid. The third column shows a function `VALID-NODES` which returns a set of nodes Y such that the corresponding validity tests in the second column are true for all $y \in Y$.

In this table, some extra notation is used:

`CHECK` is a function that uses the original validity test, i.e. `CHECK(x, Y)` for a set of nodes Y , will perform the validity test for x and all $y \in Y$ and return those nodes that fulfil the test.

Ξ_x is used to refer to the children of a node x .

$X_{N|\Xi|\Pi}$ is used to refer to those nodes that can be reached from node x by following neighbours (N), children (Ξ) or parents (Π). E.g. $X_{N\Xi}$ are those nodes reachable by following the neighbours and children from node x .

N_{Π_x} means the union over the neighbours of the parents of x , i.e. $\bigcup_{\pi \in \Pi_x} N_\pi$.

A justification of the nodes returned by `VALID-NODES` will now be provided.

InsertU

Validity Test 1 Since $V \setminus X_N$ is the set of nodes not connected to node x via undirected arcs, it is obvious that there is no undirected path between them and hence they pass the test. For the second set of nodes `CHECK($x, N_{N_x} \setminus N_x^{+x}$)` it should be noticed that the only nodes that *could* be valid are the neighbours of neighbours of x . This is because only these nodes could have shared

neighbours with x , as required in the validity test. $N_x \cup \{x\}$ are deleted from this set as an edge cannot be inserted from x to x or to a neighbour of x .

Validity Test 2 If x has no parents, then this test returns all those nodes that also have no parents, trivially being true. Where x has parents, only those nodes that are children of x 's parents could possibly be valid, i.e. they must have at least x 's parents as their own parents.

DeleteU Only those nodes that are neighbours of x have an arc that could possibly be deleted.

InsertD

Validity Test 1 $V \setminus X_{N\Pi}$ gives all those nodes for which there is no semi-directed path to x . These nodes trivially satisfy the test. Similar to Validity Test 1 in InsertU, with $\text{CHECK}(x, N_{\Pi_x} \setminus \Pi_x)$, the only nodes that could be satisfied are those that are neighbours of the parents of x . This is because only these nodes could make the set $\Omega_{x,y}$ non-empty and hence possibly block the semi-directed paths to x , as required in the validity test. Π_x are deleted from this set, as they trivially fail the test.

Validity Test 2 $V \setminus X_{N\Pi}$ gives all those nodes for which there is no semi-directed path to x . Because of this, these nodes cannot be neighbours of the parents of x , otherwise there *would* be a semi-directed path to x . Therefore $\Omega_{x,y}$ is empty and so the test is trivially satisfied. Similar to the validity test directly above, only the neighbours of the parents of x could possibly make the set $\Omega_{x,y}$ non-empty and hence fail the test.

Validity Test 3 If x has no parents, then this test returns all those nodes that *have* no parents, trivially being true. Where x has got parents, any node except for x and the parents of x could possibly pass this test.

DeletedD Trivially, only those nodes that are children of x could possibly be valid.

ReverseD Trivially, only those nodes that are children of x could possibly be valid, so these are checked for both validity tests.

MakeV As the validity test is exactly the same as Validity Test 1 of InsertU, the same idea prevails. The only difference is that as a consequence of the operation, x and y are necessarily connected, there is no need for the $V \setminus X_N$ term; if this was included,

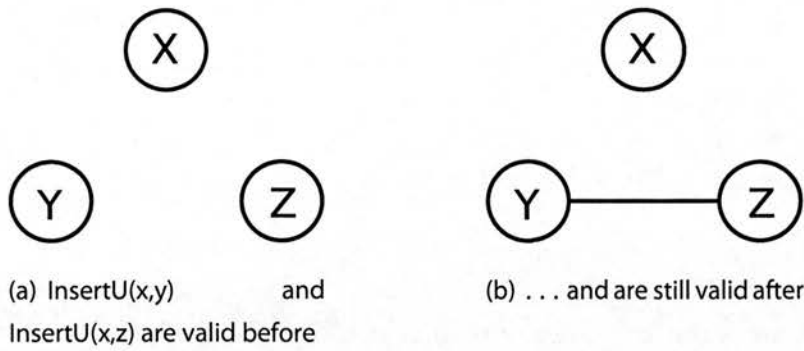


Figure 4.1: Validity of moves not changing

there would be no node z such that it is a neighbour of x and y and a member of $V \setminus X_N$.

From looking at Table 4.3, it can be seen that the number of validity checks for a given node x is now bounded by a number less than n , where n is the number of nodes on the graph. E.g. N_{N_x} is the set of neighbours of neighbours of x . Therefore, in this case, the number of checks is bounded by the number of nodes that are of distance 2 from x . If the number of parents, children and neighbours that a node can have is given an upper bound k (as is normally the case) then there are at most k^2 checks. This means the number of times an operator now needs to be checked is in $O(nk^2)$ as opposed to $O(n^2)$. This behaviour should lead to a speed up of validity checking, especially for large values of n .

4.2.2 Caching

Looking again at the behaviour of a search through the problem space, it can be seen that most moves affect only a subset of the nodes V . As an example consider Figure 4.1. If node x is not connected to node y or z then adding an undirected arc between y and z will not affect the validity of adding an undirected arc from x to y or z . This behaviour can be taken advantage of, by caching the values of validity tests for particular moves. Such a caching system would contain, e.g.,

- InsertU(x, y), InsertU(x, z) and InsertU(y, z) before the move above was made; and
- InsertU(x, y) and InsertU(x, z) after the move was made.

There are four different situations that such a caching systems would need to handle:

- Valid move is still valid;
- Valid move is no longer valid;
- Invalid move is now valid; and
- Invalid move is still invalid.

There are many ways that these semantics could be handled, e.g. the cache could be completely dumb and remove all moves and recompute the valid moves at each stage. Or, it could be more selective, and possibly only remove some entries that are still valid. Because making a move in one part of the graph can influence possible moves in a far away part of the graph, care needs to be taken in handling the cache entries.

4.2.2.1 A Caching Algorithm

In order to put the ideas above into practice an algorithm needs to be specified that works according to their principles. An algorithm that performs the necessary operations on a caching system is given in Algorithm 4.1. This algorithm, UPDATE-CACHE, is fairly

Algorithm 4.1 UPDATE-CACHE

Input: PDAG $\mathcal{P}^{new}, \mathcal{P}^{old}$, Operators O , Cache $Cache$

Output: Cache $Cache$

```

 $C \leftarrow \text{CHANGED-NODES}(\mathcal{P}^{new}, \mathcal{P}^{old})$ 
for each operator  $o \in O$  do
  for each changed node  $c \in C$  do
     $Check \leftarrow Check \cup \text{CHECK-NODES}(o, c, \mathcal{P}^{old})$ 
     $Check \leftarrow Check \cup \text{CHECK-NODES}(o, c, \mathcal{P}^{new})$ 
  end for
  for each node  $x \in Check$  do
     $Cache \leftarrow Cache \setminus \text{CACHE-VALID}(o, x, Cache)$ 
     $valid \leftarrow \text{VALID-NODES}(o, \mathcal{P}^{new}, x)$ 
     $Cache \leftarrow Cache \cup \{o, x, valid\}$ 
  end for
end for
return  $Cache$ 

```

conservative in its nature, as it tends to delete quite a lot of valid entries that are still valid. However, it is quite simple in its operation. An explanation of the algorithm is as follows.

Operator	CHECK-NODES
InsertU	X_N
DeleteU	$N_x \cup \{x\}$
InsertD	$X_{N\Xi}$
DeleteD	$N_x \cup \{x\}$
ReverseD	$X_{N\Pi}$
MakeV	X_N

Table 4.4: The nodes that must be checked for a change at node x

The algorithm receives as input:

- the PDAG that has been modified by the last move in the search space \mathcal{P}^{new} ;
- the PDAG before this modification took place \mathcal{P}^{old} ;
- the set of operators being used O ; and
- a set of cached validity tests *Cache*.

The algorithm returns the modified cache *Cache* at the end of the procedure. Firstly, UPDATE-CACHE calls the CHANGED-NODES function. This calculates the set of nodes C , such that any edges incident to each $c \in C$ have changed from \mathcal{P}^{old} to \mathcal{P}^{new} . This set cannot be calculated from the previous move taken, as in performing a single operation, cascading changes can occur from one PDAG to the next. Next, for each operator o and each node $c \in C$ associated with a change, the other nodes that might have been affected by that change are identified by the CHECK-NODES procedure. The value for this is calculated differently for each operator – Table 4.4 gives values for each of Chickering’s six operators. As an example of why this is the case, consider Figure 4.2. It is possible to add an undirected path between x and w , until an arc is added between y and z , even though both x and w are far away from the added arc in the graph. In Figure 4.2, y and z would be elements of the set C returned from CHANGED-NODES. Therefore from Table 4.4, both x and w would be returned from CHECK-NODES. As has been mentioned, the algorithm is quite conservative in removing entries from the cache. The procedure CACHE-VALID removes all cache entries with each $x \in Check$ as the first argument for

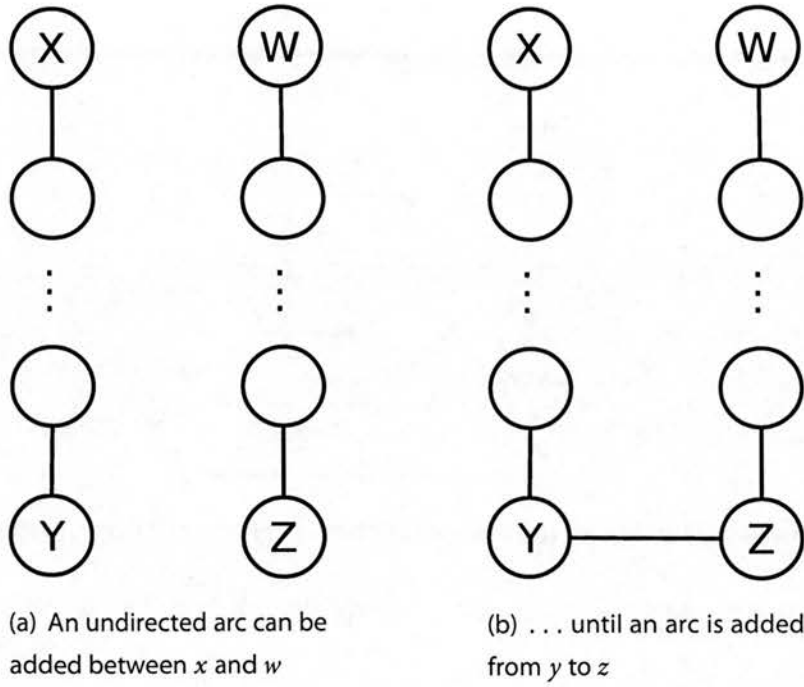


Figure 4.2: Adding edges can affect nodes far away

each operator $o \in O$. Next, the new set of valid moves are calculated as the set of nodes that is valid given the operator o , PDAG \mathcal{P}^{new} and first argument node x . The calculation of this set of valid moves is performed as specified in Table 4.3. Finally the new set of valid nodes are entered into the cache.

The correctness of this algorithm is quite easily seen. The set of nodes *Check* that is calculated, consists of all the nodes that could *possibly* be affected by a particular move. Any move involving these nodes is removed from the cache, which ensures that no invalid moves are kept in the cache. The set of valid moves is then calculated anew, with the correctness of these moves being shown in Section 4.2.1. This ensures that all valid moves are in the cache.

It is hard to quantify the effect of the caching operations on the complexity of operator validity testing. In any event, it is likely to lower the amount of nodes checked from n . Informally it was noticed that this caching procedure works best when the graph is not fully connected, e.g. when the graph is empty, almost no new checks need be done. And although the actual effect might be hard to characterise, the worst-case complexity is somewhat easier to see.

The worst-case complexity depends on the operators O that are passed into the algorithm. However, assuming the standard operators given in Table 4.3, the following

analyses can be given.

In all cases, the CHANGED-NODES function needs a search over both sets of edges and therefore is in $O(E)$. The maximum number of nodes returned from the CHANGED-NODES function can be assumed to be V . The CHECK-NODES function is in $O(V + E)$ and can be assumed to return V in the worst case. If the cache is structured as a hash map, the removing and adding the entries can be achieved in constant time. This leaves the function VALID-NODES. At the end of Section 4.2.1 it was seen that at most k^2 checks were needed for each node. Each of these checks is in $O(V + E)$ in the worst case. Therefore, the complexity for each node is in $O(k^2V + k^2E)$. Therefore, for all of the nodes in the graph, the complexity is in $O(k^2V^2 + k^2EV)$. For a fully connected graph, E is in V^2 . Therefore, the total worst case complexity for the UPDATE-CACHE algorithm is in $O(k^2V^3)$.

4.2.2.2 Implementation Issues

Although a caching algorithm can speed up move validity testing, it comes at the price of having to store this information in memory. For reasonable sized data sets, this might be an acceptable price, but for larger sizes, this procedure might not be feasible. Consider the InsertU operator from Table 4.2. With an empty graph, there are roughly $n^2/2$ valid moves, each of which would be entered into the cache. For 10,000 variables there would be $10,000^2/2 = 50,000,000$ moves. Assuming it is possible to store the operator and two arguments in a 32 bit word, this is roughly 200MB for each operator, bearing in mind that not all operators are valid at all times. Still, this gives an idea of how many entries the cache could store; anything above the limit would have to be calculated as needed.

4.3 Using Ant Colony Optimisation in Learning an Equivalence Class

To date, many state-based search algorithms that create a Bayesian network structure have relied on simple techniques such as greedy-based searches. These can produce good results, but have the ever prevalent problem of getting caught at local minima. More sophisticated heuristics have been applied, such as iterated hill climbing and simulated annealing (Chickering et al., 1995), but so far, none of these have been applied to E-space. A related approach, by Acid and de Campos (2003) applied tabu search to a space of restricted partially directed acyclic graphs (RPDAGs), a halfway house between the

spaces given by DAGs and CPDAGs.

This section seeks to apply the ACO metaheuristic to E-space, the space of equivalence classes of DAGs. To this end, two novel extensions are made to the basic metaheuristic. The first is to allow multiple *types* of moves. This is to allow more than one operator to be used in traversing the state space. This is needed, because in general, more than one type of operator is used whilst searching in E-space. The second is to allow the pheromone to be accessed (indexed) by arbitrary values – normally it is accessed by a single numerical index or two indices. Again this is needed because of the operators used in E-space – the MakeV operator takes three nodes as arguments.

The proposed algorithm, ACO-E, is a continuation of the work by Daly et al. (2006b) and Daly and Shen (2008), and is based in large part, on the work of de Campos et al. (2002a). In that work, an ACO algorithm called ACO-B was applied to learning Bayesian networks. This current work differs in that it searches in E-space, uses more than one operator (most other ACO applications use a single operator) and does not constrain itself to using matrices to store pheromone. The algorithm is shown in Algorithm 4.2.

4.3.1 Relation of ACO-E to the ACO Metaheuristic

In this section, the relation of the various parts of the algorithm to the ACO framework will be given. The problem of learning a Bayesian network structure can be stated as the triple (\mathcal{S}, f, Ω) , where

- \mathcal{S} , the set of all candidate solutions, is the set of all CPDAGs on the nodes of the Bayesian network. This set has a massive cardinality, being super-exponential in the number of nodes.
- f , the objective function, is the function used to score a candidate CPDAG. This function would generally be one of the scoring criteria mentioned in Section 4.1 and in Section 2.4.6.
- Ω , the set of constraints, makes sure that only PDAGs that have consistent extensions are generated as solutions. An explanation of the idea of a consistent extension of a PDAG is given in Section 2.1.3. In the formulation being presented, the constraints are implicit in the operators that will be used to move from state to state.

Given this statement of the problem, the ACO-E algorithm can be described by the following properties. These properties relate to the ACO metaheuristic described in

Section 3.2.

4.3.1.1 The Construction Graph

The construction graph in an ACO algorithm describes the mechanism by which solutions can be assembled. It is specified as the complete graph given over the solution components. As such, these components play a crucial part in the viability of the algorithm. An example construction graph is shown in Figure 4.3. It should be noted that this construction graph is partial – nodes for the DeleteU and DeleteD operators are missing and certain arcs connecting nodes are missing. Note that in this case, no InsertD operators would be present. It should be noted that this graph represents *all* possible moves at the start of the run – after certain moves, certain arcs would not be traversable. E.g. after going to the three InsertU moves after the start, none of the MakeV moves would be accessible.

In the ACO-E algorithm, the components (nodes) C of the construction graph are the various moves that may be made, i.e. each *move* is an *instantiation* of a supplied operator; in the experiments presented in this thesis, the six operators in Table 4.2 are used. These operators are used as they have been verified to work correctly and effectively by Chickering (2002a). Designing correct operators is difficult, as Chickering showed by finding counter examples to the operators of Munteanu and Cau (2000). Each ant constructs a solution by walking the construction graph. This corresponds to applying a sequence of moves to a CPDAG. In order for the procedure to begin, a starting state must be specified. In ACO-E this is given as the empty graph.

As usual, the states of the problem are sequences of moves. However, because every state is a candidate solution, $S = \mathcal{X}$ in the ACO metaheuristic framework. This does not imply that all states are *feasible* candidate solutions, but only that candidate solutions can be of any length. This also means that $\tilde{S} = \tilde{\mathcal{X}}$. Another way to view the state of an ant is to consider the empty graph \mathcal{P} (the starting state) and the current state as a sequence of moves (components) $x = \langle c_i, \dots, c_j \rangle$. Applying each move $c \in x$ in order to \mathcal{P} will give a CPDAG that is another representation of the current state.

It should be noted that the constraints Ω are implicitly taken care of by the operators, i.e. the validity tests on the operators satisfy the constraint that each state is a valid PDAG. It should also be stated that the usual definition of

$$g(s, t) \equiv f(s, t), \forall s \in \tilde{S}$$

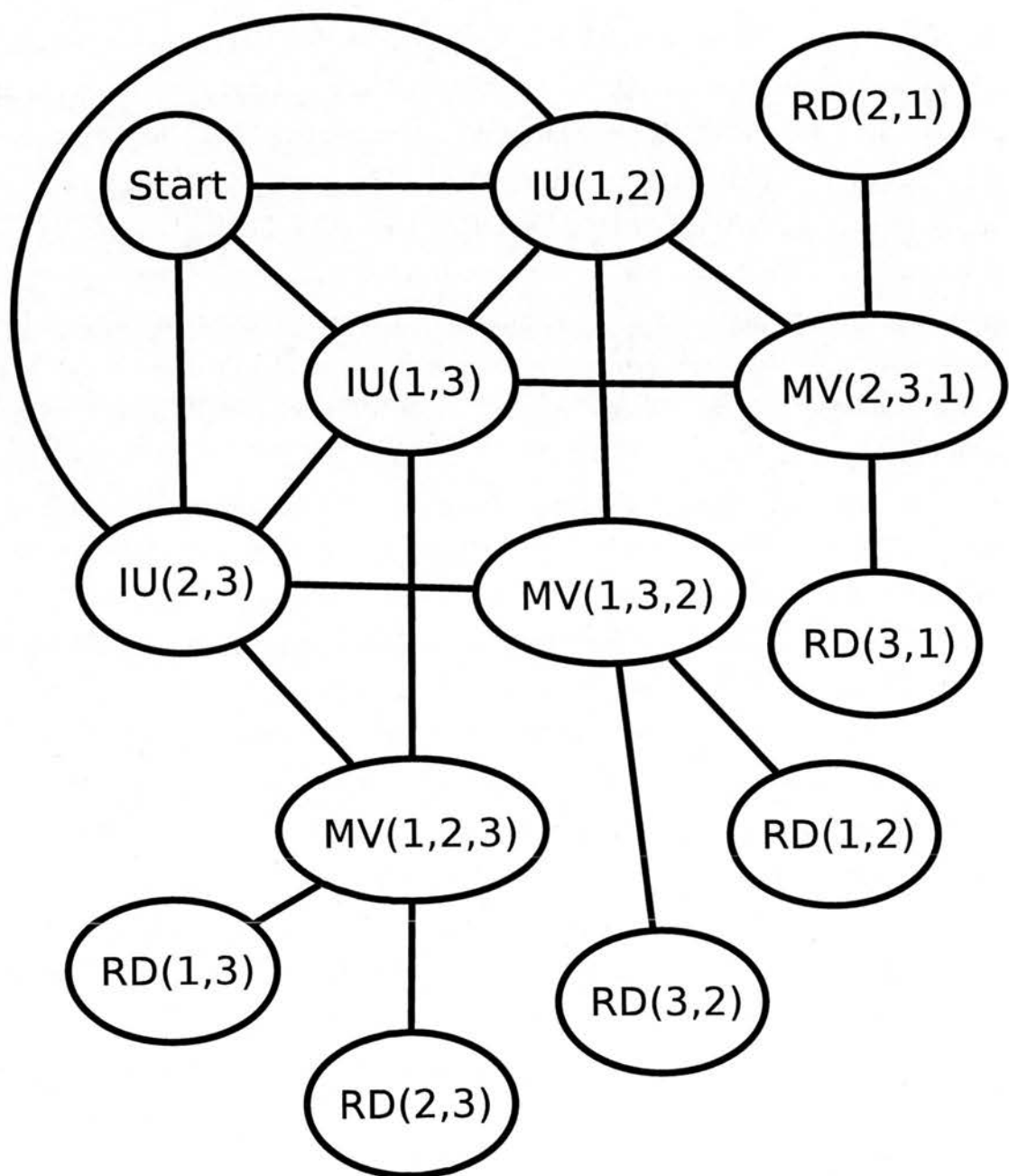


Figure 4.3: An example partial construction graph for the three node network shown in Figure 4.4

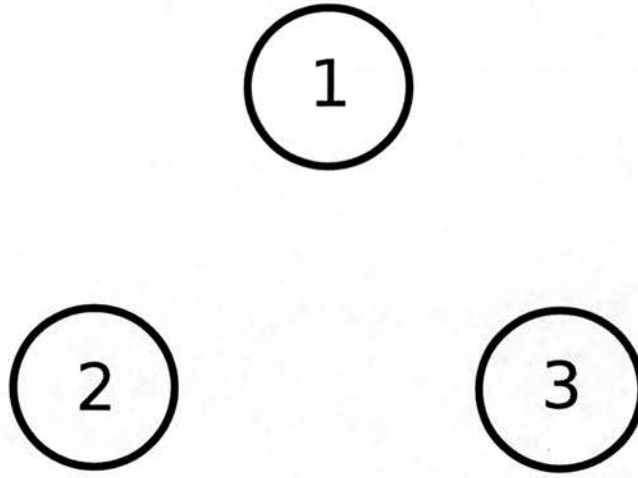


Figure 4.4: A three node network used for illustration by Figure 4.3

applies, and that there is no function $J(x, t)$. This is because all x are candidate solutions (and J is normally only needed for states that are not candidate solutions) and adding a solution component can decrease the cost (and J should monotonically increase with each component added).

4.3.1.2 The Problem Heuristic

In an ACO algorithm, the heuristic is used to guide the search to good solutions. It often does this implicitly in terms of a cost associated with choosing a particular component to add to the current state; adding a component with the least cost is often a good way of proceeding in constructing a solution.

In ACO-E, the heuristic is used in the same manner, with the addition that the cost for adding a component can be negative, i.e. adding a component to the current state can improve the cost function g . The heuristic is dynamic in that it depends on the current state of the ant. Also, it is associated with each component (node) $c \in C$ as opposed to the arcs $c_i - c_j$ between components.

The value of the heuristic η_i is given by the score gain for each move $c_i \in C$ that is possible given the current state. In essence it corresponds to the change in score given by performing a particular move on the current CPDAG. For the operators being used in this thesis, this means the change in score given in Table 4.2.

4.3.1.3 The Problem Pheromone

The pheromone in an ACO algorithm guides the search based on the results of previous ants' searches. In many instances, it is associated with the arcs on the construction graph, but in ACO-E it is associated with the nodes of the construction graph. This gives pheromone values τ_i for each $c_i \in C$.

The pheromone for each τ_i is initialised to a value τ_0 given by

$$\tau_0 = \frac{1}{n |\text{SCORE}(\mathcal{P}^+)|}. \quad (4.3)$$

In this formula, n is the number of variables that are in the data, SCORE is the objective function f , as defined in Section 4.3.1, and \mathcal{P}^+ is the best-so-far solution. At the start of the algorithm, this is initialised to that found by a greedy search starting from the empty graph.

In order that the pheromone may change to reflect the foraging of ants, pheromone update rules are given. Similar to ACS, there is a local evaporation rule, whereby there is a net removal of pheromone from a path as an ant traverses it

$$\tau_l^{t+1} \leftarrow (1 - \rho) \tau_l^t + \rho \tau_0$$

This shows the effect of the parameter ρ , which is the pheromone evaporation and deposition rate. With this formula, there are implicit bounds on how high and low the pheromone at each component can get. Note that it is not strictly necessary to have the evaporation and deposition rate be the same parameter – in some formulations of ACS there is a separate parameter ξ that controls evaporation. However in ACO-E, these parameters were constrained to be the same value, so as to minimise the number of parameters needed. Also similar to ACS, there is a global pheromone update rule that deposits new pheromone on the best-so-far path

$$\tau_l^{t+1} \leftarrow (1 - \rho) \tau_l^t + \rho / |\text{SCORE}(\mathcal{P}^+)|$$

This occurs at the end of a run of ants. Again, SCORE and \mathcal{P}^+ are defined as in equation 4.3. Also again, this formula implements implicit limits on the values that pheromone can take.

4.3.1.4 Probabilistic Transition Rule

In choosing which component to visit next given a particular state, an ACO algorithm utilises a probabilistic transition rule. This rule normally uses values given by the heuristic

and pheromone to inform the choice of which node to pick. The actual choice is random and is based on a distribution given by the heuristic and pheromone of each possible choice. In ACO-E, the probabilistic choice rule is given by a pseudo random proportional choice rule, very similar to the one used in ACS. This type of rule allows the balance between exploration and exploitation to be varied. Being able to change this balance is important, as it has been shown to produce quite different results (Dorigo and Stützle, 2004). An ant chooses component c_l , where l is given by

$$l \leftarrow \begin{cases} \arg \max_{l \in \mathcal{N}(x)} \tau_l [\eta_l]^\beta, & \text{if } q \leq q_0 \\ \text{random proportional,} & \text{otherwise} \end{cases}.$$

In this formula, $\mathcal{N}(x)$ is the set of components that an ant at state x can move to, given the problem constraints Ω . The rule is *pseudo*-random proportional, because it sometimes behaves in a manner that is not random. A random number q is drawn uniformly in the range $[0, 1]$. If this number is less than or equal to a parameter q_0 , then the rule behaves greedily; the best move possible is taken dependent on the value of $\tau_l [\eta_l]^\beta$ for each component c_l . Here, τ_l and η_l are the pheromone and heuristic as explained previously and β is a parameter that says how much to favour the heuristic over the pheromone.

If the number q is greater than q_0 then a random proportional rule is used to select which component to visit next. The probability that the ant will visit component c_l is given by p_l , where

$$p_l = \frac{\tau_l [\eta_l]^\beta}{\sum_{\mu \in \mathcal{N}(x)} \tau_\mu [\eta_\mu]^\beta}, \quad \forall l \in \mathcal{N}(x). \quad (4.4)$$

It can be seen that the probability that an ant moves to component c_l is directly given by $\tau_l [\eta_l]^\beta$, normalised over the other possible moves so that it is in the range $[0, 1]$.

4.3.1.5 Properties of Ants

In terms of the ants used to construct solutions, the following properties of ant k should be noted:

- The memory \mathcal{M}^k can be equated to the current state of the problem given by ant k . From this, the current CPDAG can be constructed in order to implement the constraints Ω , compute the heuristic values η , evaluate the current solution and lay pheromone on the tour. In practice, the current CPDAG is normally kept in order to avoid having to recompute it at every step.

- The start state x_s^k is given by the empty sequence $\langle \rangle$, i.e. the empty CPDAG.
- The single termination condition e^k , is to stop the ant's traversal of the construction graph, when no improvement in score is possible.
- The neighbourhood $\mathcal{N}^k(x)$ is the set of all valid moves given the current CPDAG and node x .

4.3.1.6 Local Search Procedure

As is often the case with ACO algorithms, ACO-E can use a local search procedure at intermediate points throughout the run of the algorithm and at the end. This local search procedure can be used to quickly bring a solution to a local maximum. With the current heuristic and the standard local search that would be used in these circumstances – greedy search with the operators defined in Table 4.2, known here as GREEDY-E – local search would provide no additional benefit over the solution found by an ant. Nevertheless, the local search was put in the algorithm in case the problem heuristic were to be implemented differently. An example of this would be a static heuristic obtained by scoring operations on an empty graph. Since this is invariant over the algorithm run, it would only need to be calculated once at the start of the run.

4.3.2 Description of ACO-E

This section will focus on giving an algorithmic description of ACO-E. This is done in conjunction with the pseudo code given in Algorithms 4.2 and 4.3. ACO-E takes as input a number of parameters and returns the best PDAG found according to a scoring criterion, SCORE, that is defined as the objective function f . It is assumed that scoring criteria generally give negative values; the higher the value, the better the model. This is the case of most of the standard criteria as discussed in Section 2.4.6. The meaning of the parameters is as follows:

- O This is a set of operators that can modify the current PDAG state in the search. Examples of these are the ones given in Table 4.2, e.g. InsertU, DeleteU, etc. However, other operators could be used, e.g. those of Munteanu and Cau (2000) and Munteanu and Bendou (2001).
- t_{max} This is the number of iterations of the algorithm to run. At each iteration, a number of ants construct solutions. Pheromone deposition happens after all the ants have finished their walks.

t_{step} This is the gap, in iterations, between which local search procedures are run. If set so that $t_{step} > t_{max}$, then local search only happens at the end of the algorithm run.

m This is the number of ants that run at each iteration.

ρ This, a value in $[0,1]$, is the rate at which pheromone evaporates and is deposited. It is used in both the pheromone evaporation and pheromone deposition rules in Section 4.3.1.3.

q_0 This, a value in $[0,1]$, gives the preference of exploitation over exploration. It is used in the pseudo-random probabilistic transition rule as explained in Section 4.3.1.4.

β This exponent gives the relative importance of the heuristic over the pheromone levels in deciding the chance that a particular trail will be followed. It is used in the pseudo-random probabilistic transition rule in Section 4.3.1.4.

n This is the number of nodes in the PDAG.

There are also other variables in the algorithm. These include:

\mathcal{P}^+ the best-so-far PDAG;

$Path^+$ the best-so-far path;

\mathcal{P}^{empty} the empty PDAG; and

$Path^{empty}$ the empty path, i.e. the path with no entries.

In starting the algorithm, a greedy search (called GREEDY-E) is performed. This is a search through the space of equivalence classes using the framework and operators given by Chickering (2002a) and shown in Table 4.2. It gives a starting best-so-far graph and path from which the search can proceed. Pheromone levels for each solution component are then initialised to τ_0 . The main loop of the algorithm then begins for t_{max} iterations. At each iteration, m ants perform a search, given by algorithm ANT-E, shown in Algorithm 4.3. Also, for every t_{step} iterations, a local search is performed on the PDAGs returned from ANT-E, to try and improve results. Using local search as part of an ACO algorithm is a very common technique (Dorigo and Stützle, 2004), as it is an easy way to obtain good results with little effort. After the m ants have traversed the graph, the best graph and path are selected from the best-so-far graph and path and the

ones found by each of the ants in the current iteration. Finally, the global pheromone update lays pheromone on the best-so-far path.

The ANT-E algorithm creates a PDAG by examining the various states that may be proceeded to from the current state, given a set of operators that may act on the current PDAG. It then selects a new state based on a random-proportional choice rule. The parameters to the function have the same description as the ones to the ACO-E function.

Starting out, the algorithm constructs an empty PDAG. Then at each stage a move is made to a new PDAG, which can be reached by applying one of the operators in O . Initially, a number is given to each move by *TOTAL-SCORE*, shown in Algorithm 4.4. This number represents a weight given to each move l depending on the current pheromone associated with making that move τ_l , and the heuristic associated with making the move η_l . This heuristic is given by the increase in score obtained by taking that move, higher overall scores meaning better solutions. If there can be no increase in the score, the ant stops and returns the solution \mathcal{P} and the path followed. Otherwise there is a possible move and the ant decides how to make it. Firstly a random number q is obtained. If it is less than a specified value q_0 , then the best move is taken. If it is greater than q_0 , then a random proportional choice is made, with the probability of better moves being higher. After this, a local pheromone update is applied to the path just taken, the path is updated with the new location at the end and the current state is updated to become the new state given by l . Note that applying a move to a CPDAG to change state implies that the resulting PDAG will be extended to a DAG by a suitable method (e.g. that of Dor and Tarsi, 1992) and this DAG be changed back to a CPDAG. Details can be found in (Chickering, 2002a).

4.3.3 Implementation Issues

In implementing the algorithms given in this section, care must be taken to avoid long run times. The easiest method to speeding up any search for a Bayesian network structure is to cache the score of a node given its parents. Secondly, the methods of Section 4.2, which involve reducing the number of validity tests which must be performed and caching the results of these tests, can again increase performance dramatically. These methods were designed for algorithms such as ACO-E, which use multiple restarts from the same starting state.

Care must also be taken in implementing the pheromone for the moves. Traditionally, matrices of values are used, which allow fast access and updating. However in the case

of the MakeV operator, which takes three indices, a three dimensional matrix would be needed. This would quickly become infeasible as the problem size grew, especially as only some of the entries would ever be used due to the algorithm never getting to those states. Instead a structure such as a map can store this information. A map can scale linearly with the number of elements actually being used. If the map is implemented as a tree, entries can be accessed in logarithmic time and if a hash table is used, access can be in constant time. In the experiments described in Section 5.3, the map is implemented as a binary tree.

4.4 Summary

This chapter introduced methods to solve two different problems in learning the structure of a Bayesian network. The first problem addressed long running times when running algorithms with multiple restarts. The second problem dealt with trying to find the best Bayesian network structure that ‘fits’ a set of data.

To help solve the first problems two new technique were demonstrated. The first of these techniques involved reducing the number of validity tests that needed to be performed whilst performing a search in the space of equivalence classes of Bayesian networks. For each of the operators defined by Chickering (2002a), a new formula was defined to calculate the validity tests needed to see whether a move is valid. These formulae reduced the number of tests that need to be performed for each operator, from $O(n^2)$ to $O(nk^2)$, where k is the bound on the number of parents and neighbours a node can have.

The second technique involved caching the results of validity tests of a move, so that they do not need to be recalculated. An algorithm called UPDATE-CACHE was developed in order to update the cache after a move is made, in order that the entries in the cache are correct. The effectiveness of this algorithm is dependent on the connectivity of the graph; a less connected graph will tend to have better performance characteristics.

To help solve the second problem, that of finding a good Bayesian network structure, an algorithm based on ant colony optimisation (ACO) was defined. This algorithm, called ACO-E was based on the ACO-B algorithm of de Campos et al. (2002a) and the search space defined by Chickering (2002a). A description of the algorithm was given in terms of the ACO metaheuristic and high-level pseudocode.

In the next three chapters, the effectiveness of these algorithms will be tested. Chapter 5 defines a testing methodology and reports the results of experiments performed on

both of the algorithms. Chapter 6 involves a discussion of the results that were found and an interpretation of their significance. Chapter 7 will show an application of these techniques to a real-world machine learning problem.

Algorithm 4.2 ACO-E

Input: Operators O , t_{max} , t_{step} , m , ρ , q_0 , β , n **Output:** PDAG \mathcal{P}^+ $(\mathcal{P}^+, Path^+) \leftarrow \text{GREEDY-E}(\mathcal{P}^{empty}, Path^{empty})$ $\tau_0 \leftarrow 1/n |\text{SCORE}(\mathcal{P}^+)|$ **for** each operator o in O **do** **for** each possible move l in o on \mathcal{P}^{empty} **do** $\tau_l \leftarrow \tau_0$ **end for****end for****for** $t \leftarrow 1$ to t_{max} **do** **for** $k \leftarrow 1$ to m **do** $(\mathcal{P}^k, Path^k) \leftarrow \text{ANT-E}(O, q_0, \rho, \beta, \tau_0)$ **if** $(t \bmod t_{step} = 0)$ **then** $(\mathcal{P}^k, Path^k) \leftarrow \text{GREEDY-E}(\mathcal{P}^k, Path^k)$ **end if** **end for** $b \leftarrow \arg\max_{k=1}^m \text{SCORE}(\mathcal{P}^k)$ **if** $\text{SCORE}(\mathcal{P}^b) > \text{SCORE}(\mathcal{P}^+)$ **then** $\mathcal{P}^+ \leftarrow \mathcal{P}^b$ $Path^+ \leftarrow Path^b$ **end if** **for** each move l in $Path^+$ **do** $\tau_l \leftarrow (1 - \rho) \tau_l + \rho / |\text{SCORE}(\mathcal{P}^+)|$ **end for****end for****return** \mathcal{P}^+

Algorithm 4.3 ANT-E

Input: Operators O , ρ , q_0 , β **Output:** PDAG \mathcal{P} , Path $Path$

```

while true do
   $M \leftarrow$  All possible moves from  $\mathcal{P}$  using  $O$ 
  if  $|M| = 0 \vee \max_{l \in M} \text{TOTAL-SCORE}(l, \beta) = 0$  then
    return  $(\mathcal{P}, Path)$ 
  end if
   $q \leftarrow$  random number in  $[0, 1)$ 
  if  $q \leq q_0$  then
     $l \leftarrow \arg\max_{l \in M} \text{TOTAL-SCORE}(l, \beta)$ 
  else
     $l \leftarrow$  random choice according to equation 4.4
  end if
   $\tau_l \leftarrow (1 - \rho) \tau_l + \rho \tau_0$ 
   $\mathcal{P} \leftarrow$  apply  $l$  to  $\mathcal{P}$ 
   $Path \leftarrow$  append  $l$  to  $Path$ 
end while

```

Algorithm 4.4 TOTAL-SCORE

Input: Move l , β **Output:** Score s

```

return  $s$  such that  $s = \begin{cases} \tau_l (\eta_l)^\beta & \text{if } \eta_l > 0 \\ 0 & \text{otherwise} \end{cases}$ 

```

Chapter 5

Experimental Methodology

THIS CHAPTER is concerned with testing the algorithms presented in Chapter 4 and the evaluation of the results produced. In order to facilitate understanding of the experimental methodology used, the chapter will be structured as follows.

Firstly, an account will be given of the objects on which the testing will be performed. These objects are six gold-standard Bayesian networks that are well known in the field. The various properties of the networks, along with a visual representation in the form of their DAG structure, will be discussed. From these networks, data can be sampled using a procedure that will be described and it is this data that can be used as input to the algorithms.

Next, an explanation of experiments involving the methods presented in Section 4.2 will be given. This will involve a discussion of the methodology used to run the experiments and the evaluation criteria that will be used.

Finally, experiments using the ACO-E algorithm described in Section 4.3 will be shown. As before, the methodology used in running the experiments will be defined, along with a description of the various evaluation criteria. These involve criteria well known in the field, along with criteria developed as a result of this thesis. Two different sets of experiments will be presented, one focused on the comparison of ACO-E against similar algorithms, the other a comparison of ACO-E against state-of-the-art algorithms. Also, the behaviour of the ACO-E algorithm for different parameters will be shown.

5.1 Standard Bayesian Networks

In this section a set of six gold-standard Bayesian networks will be presented. These networks will be the basis of the testing that will be showcased later. Various properties of the networks will be given, covering:

- the number of nodes of the structure;
- the number of edges in the structure;
- the average number of incoming edges;
- the average number of outgoing edges; etc.

Also, diagrams of the various networks will be presented in order to gain an appreciation of their topology and how this might affect the performance of the algorithms. To test the algorithms that have been presented, data needs to be sampled off these Bayesian networks. The procedure that is used to perform this sampling will be illustrated.

5.1.1 Six Gold-Standard Networks

In the experiments shown in this chapter, six gold-standard networks are used. These are the ALARM (Beinlich et al., 1989), Barley (Kristensen and Rasmussen, 2002), Diabetes (Andreassen et al., 1991), HailFinder (Abramson et al., 1996), Mildew (Jensen, 1995) and Win95pts networks (Microsoft Research, 1995). These networks were chosen because they covered a wide range of domains, were easily available and all contained discrete attributes. The last property was important because the scoring criterion that has been used in the experiments is implemented over multinomial random variables.

Figures A.1 to A.6 in Appendix A show the structure of the Bayesian networks in question. Various properties of these Bayesian networks are shown in Table 5.1. In this table, *Nodes* and *Edges* specify the number of nodes and edges respectively in the graph. *Probabilities* gives the total number of probabilities that are specified in the conditional probability tables over all the nodes. The *Mean In-Degree* is the average number of arcs coming into a node in the graph. This is equal to the Mean Out-Degree and the number of edges divided by the number of nodes. The *Max In* and *Out-Degree* specify the largest in and out-degree respectively of all the nodes in the graph. *Mean* and *Max Outcomes* specify the mean and maximum number of possible outcomes (values a variable can take on) over all the nodes in the graph. Finally, *V-Structures* and *V-Struct/Nodes* show

	Alarm	Barley	Diabetes	HailFinder	Mildew	Win95pts
Nodes	37	48	36	56	35	76
Edges	46	84	48	66	46	112
Probabilities	752	130180	32705	3741	547158	1148
Mean In-Degree	1.24	1.75	1.33	1.18	1.31	1.47
Max In-Degree	4	4	3	4	3	7
Max Out-Degree	5	5	5	16	3	10
Mean Outcomes	2.84	8.77	11.25	3.98	17.6	2
Max Outcomes	4	67	21	11	100	2
V-Structures	26	66	21	37	37	135
V-Struct/Nodes	0.70	1.38	0.58	0.66	1.06	1.78

Table 5.1: Bayesian network properties

the amount of v-structures in the graph and the amount of v-structures divided by the number of nodes. Following on from this, more detailed information about the distribution of nodes into groups depending on their in-degree is given in Figures 5.1 to 5.6. In these figures each column represents the number of nodes in a graph with the specified in-degree, i.e. the number of incoming arcs.

5.1.2 Sampling Data from a Network

In order to test the algorithms, data needs to be generated from the Bayesian networks being used in the study. There are many methods used for sampling data from Bayesian networks; examples are given in Section 2.2.9.1. However most of these methods are designed for dealing with inference in Bayesian networks, which involves sampling with evidence.

For the problem of generating samples for learning purposes, the complexity of these algorithms is not needed, as no evidence is specified. Therefore the original technique of Henrion (1988) can be used, knowing that samples will not be discarded as there is no evidence to match them against. The problem of generating samples can therefore be easily done; pseudocode for the algorithm is shown in Algorithm 5.1. In essence, this algorithm generates values starting at the root nodes of a Bayesian network and proceeding forward. At each step, it is possible to calculate the probability distribution for a certain node, as the parents have already been specified. With the distribution in hand,

Algorithm 5.1 Generating samples from a Bayesian network**Input:** Bayesian Network \mathcal{B} , Number of Samples N , Number of Nodes n **Output:** Data D

```

 $O \leftarrow$  Total ordering of the nodes in  $\mathcal{B}$ 
for  $i \leftarrow 1$  to  $N$  do
  Datum  $d$ 
  for  $j \leftarrow 1$  to  $n$  do
    Node  $x \leftarrow O_j$ 
    Set of nodes  $px \leftarrow \Pi_x$ 
     $d.x \leftarrow$  value selected according to  $P(x, d.px)$ 
  end for
   $D \leftarrow D \cup d$ 
end for
return  $D$ 

```

a value for a node can be selected and entered into the current datum being generated.

5.2 Accelerating the Learning Process

This section contains details of the experiments performed using the methods described in Section 4.2. Firstly, the methodology used in running the experiments will be presented, including the design of the experiments themselves and an explanation of the evaluation criteria.

5.2.1 Experimental Design

In order to test the effectiveness of the methods described in Section 4.2, experiments were run to establish by how much they could speed up the runtime of algorithms over the case when the methods were not implemented. Because these methods were designed to speed up the runtime of the ACO-E algorithm, this is used as the testing algorithm for the purposes of these experiments. The six standard networks presented in Section 5.1.1 were used and for each network, 100 experiments were run. Each individual run sampled 5000 data from the network in question, using the procedure in Section 5.1.2. From this data a scoring function could be constructed, using one of the criteria described in Section 2.4.6.

For these experiments, it was decided to use the BDeu criterion invented by Buntine

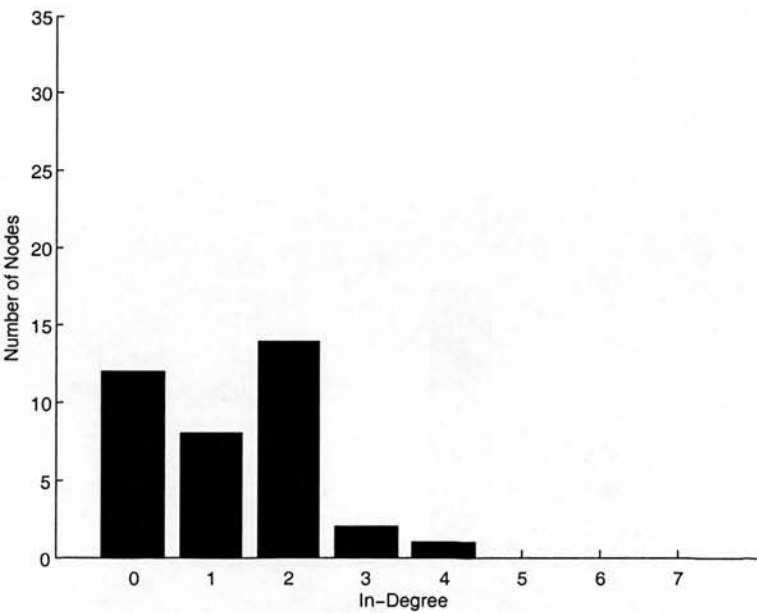


Figure 5.1: Distribution of nodes across in-degree – Alarm

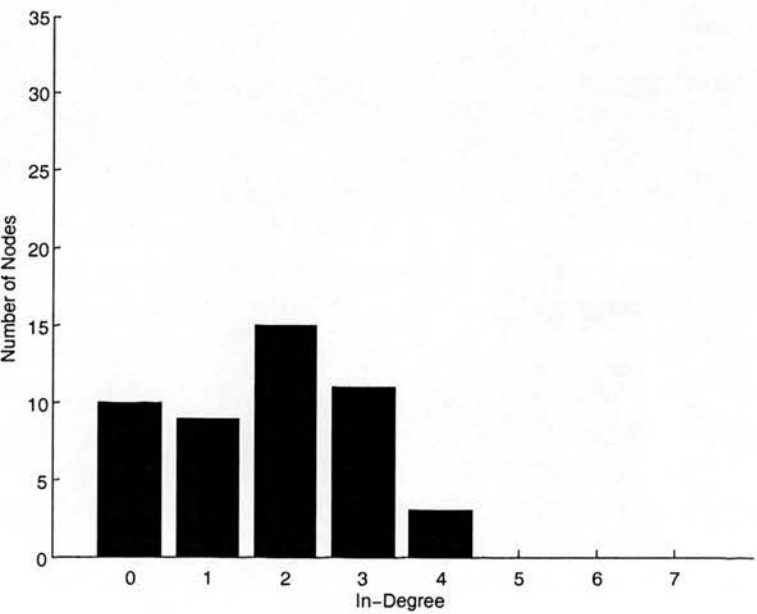


Figure 5.2: Distribution of nodes across in-degree – Barley

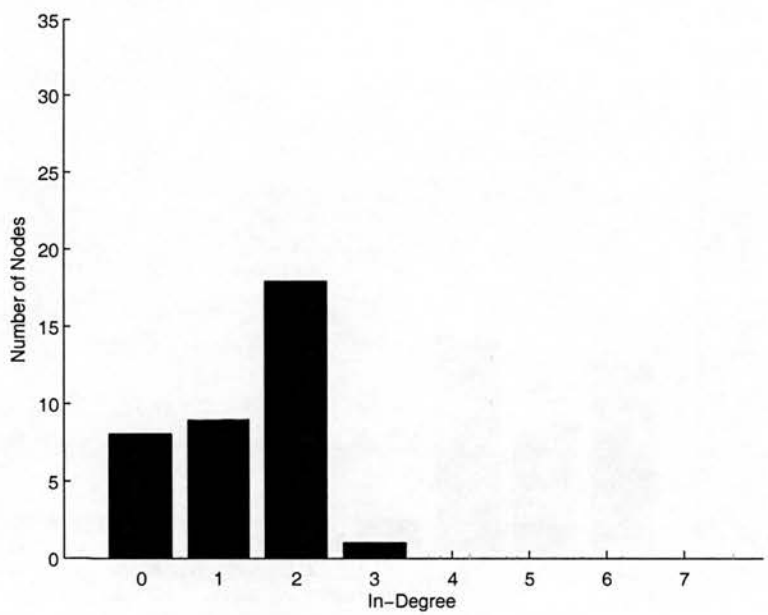


Figure 5.3: Distribution of nodes across in-degree – Diabetes

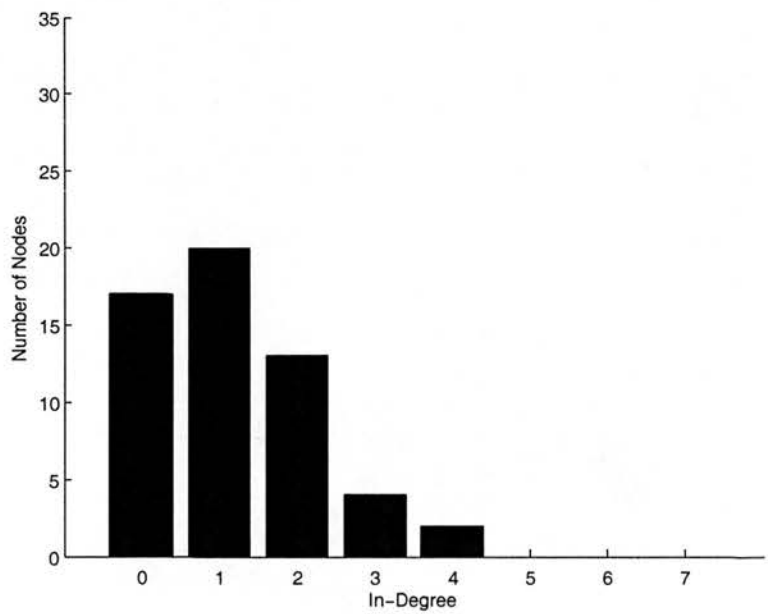


Figure 5.4: Distribution of nodes across in-degree – HailFinder

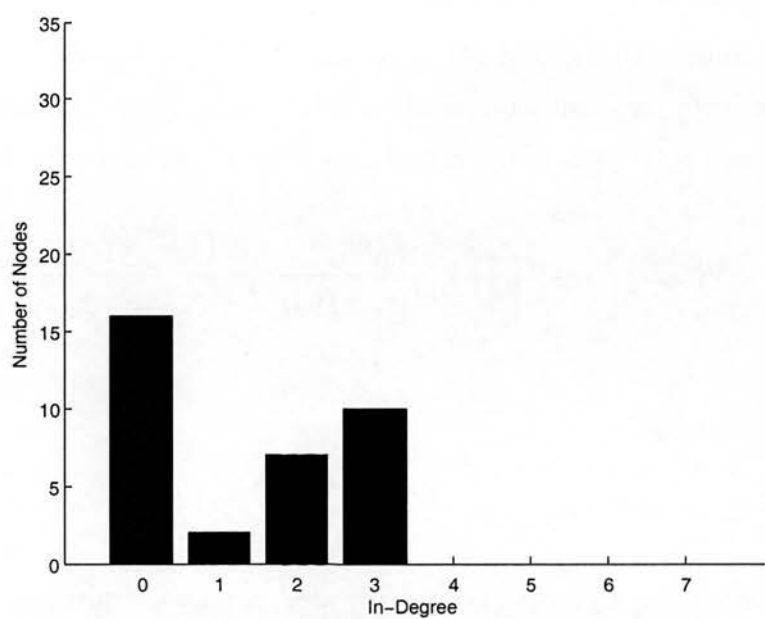


Figure 5.5: Distribution of nodes across in-degree – Mildew

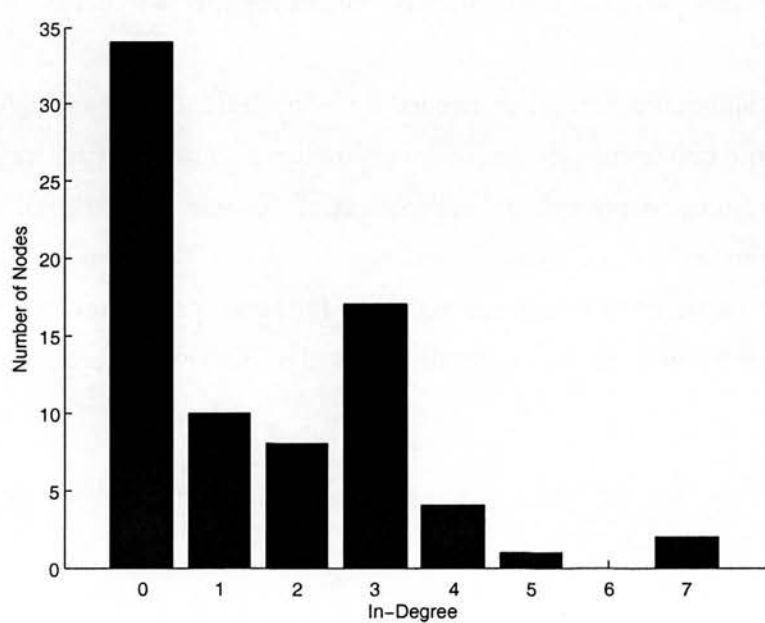


Figure 5.6: Distribution of nodes across in-degree – Win95pts

(1991) and described in Section 4.1. According to the study by Shaughnessy and Livingston (2005), BDeu had the best tradeoff between precision and recall (confusingly BDeu is called BAYES in their study, with BDeu in their study meaning the K2 metric). This criterion gives a fully Bayesian score, with the assumption of Dirichlet parameter priors and uniform parameter priors. The criterion is shown here again, in normal and log formats.

$$P(\mathcal{G}|D) = P(\mathcal{G}) \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(\frac{N'}{r_i q_i} + N_{ijk})}{\Gamma(\frac{N'}{r_i q_i})}$$

$$\log P(\mathcal{G}|D) = \log P(\mathcal{G})$$

$$+ \sum_{i=1}^n \sum_{j=1}^{q_i} \log \Gamma(N'_{ij}) - \log \Gamma(N'_{ij} + N_{ij}) + \sum_{k=1}^{r_i} \log \Gamma(\frac{N'}{r_i q_i} + N_{ijk}) - \log \Gamma(\frac{N'}{r_i q_i})$$

In implementations, the log format is normally used, as it much more numerically stable due to the multiplications and divisions being replaced by additions and subtractions, and the $\log \Gamma$ function being directly available to avoid overflow. To fully specify the BDeu criterion, two pieces of information are needed. First is a prior on structures $P(\mathcal{G})$. This could be a uniform prior, such that all structures have the same $P(\mathcal{G})$. Another method shown by Heckerman et al. (1995) was to have an expert specify a structure, and have a method that penalises differences between the expert's structure and a candidate structure.

The second piece of information needed is the 'equivalent sample size', N' , a parameter that encodes the confidence in the prior parameters and prior structure. Selecting this value can be troublesome (Steck and Jaakkola, 2002; Silander et al., 2007), but 'reasonable' values in the range $[1, 10]$ often work well.

There does not seem to be any consensus on the best type of prior to use in a Bayesian score. Apart from a uniform prior and the method of Heckerman et al. (1995), there exist methods by:

Chickering (2002a,b) that penalise structures depending on the number of free parameters in the model;

Imoto et al. (2003) that penalise structures in an iterative manner, by obtaining prior evidence from the the current hypothesis network; and

Bernard and Hartemink (2005) that penalise structures by using a p-value obtained from transcription factor binding location data to generate prior knowledge.

Parameter	N'	κ	t_{max}	m	ρ	q_0	β
Value	4	0.2	100	5	0.1	0.9	3

Table 5.2: Parameter values for testing acceleration methods

These methods generally have the property that they penalise structures with more edges than less. An obvious exception is the uniform prior, which assigns equal values to each structure. Because there are more graphs with a higher number of edges than a lower numbers of edges, there would be a bias to graphs with more edges in this case. In recognition that simpler structures are often more appealing, the prior was specified by the method shown by Heckerman et al. (1995). They specify a formula

$$P(\mathcal{G}) = c\kappa^\delta,$$

where c is a normalisation constant that can be ignored, κ is a parameter that needs to be specified and δ is given by the formula

$$\delta = \sum_{i=1}^n \delta_i,$$

where δ_i is the symmetric difference of the parent set between the current candidate structure and the prior structure for node i . An empty structure prior was specified and using the method specified in Heckerman et al., κ was set to $1/(N' + 1)$. The parameters of the scoring criterion and the ACO-E algorithm were set as in Table 5.2, with the set of operators O being set to those of Chickering (2002a) as shown in Table 4.2. All of these parameters are in the range of reasonable levels as shown by Dorigo and Stützle (2004). The above experiments were performed using two conditions:

- The methods introduced in Section 4.2; and
- Without those methods.

Each of the experiments was run on identical machines, being the sole process whilst running.

5.2.2 Evaluation Criteria

In order to evaluate the performance of the experiments, criteria needed to be established to quantify the effect produced. Reasonable criteria include:

- Recording the wall clock time in each condition; and
- Counting the number of validation checks used in each condition.

In order to give as realistic a quantification of the behaviour as possible, it was decided to record the wall clock time spent inside the programs code. This was possible, because the code base for both conditions was exactly the same, apart from the differences of the conditions themselves. It was also possible to control the machines where the experiments took place, so there would be as little effect from outside influence as possible. Also, counting the number of validation checks done would not be strictly accurate, as there are overheads to each of the methods that differ.

With this criterion selected, it is possible to plot the behaviour of the algorithm as it progressed and provide performance ratios between the two conditions.

5.3 Using Ant Colony Optimisation in Learning an Equivalence Class

This section contains details of the experiments performed using the ACO-E algorithm described in Section 4.3. Firstly, the methodology used in running the experiments will be presented. This includes an analysis of the needed outcomes, the design of two experimental conditions and an explanation of the evaluation criteria.

5.3.1 Experimental Design

In designing an experimental methodology to test the efficacy of the ACO-E algorithm, three different outcomes were desired.

- The first was to analyse the behaviour of the algorithm as a function of the parameters and the test networks. This is needed in order to try and understand the range of values in which parameters might be useful and to show the effect of the ACO behaviour on outcomes.
- The next desired outcome was to test ACO-E against other similar algorithms. To this end, ACO-E was tested against another ACO algorithm and algorithms that searched in the space of equivalence classes.

- Finally the last desired outcome was to test ACO-E against state-of-the-art algorithms from the literature. These tests would show the comparative usefulness of ACO-E against other well-known and good-performing methods.

In order to obtain these outcomes, two experimental conditions were designed.

- The first, dealing with the behaviour of ACO-E with different parameter values was used to obtain the first two outcomes.
- The second, which focused on using 'good' parameter values was used to obtain the third outcome.

5.3.1.1 Experimental Condition 1

Experimental condition 1 was designed to analyse the behaviour of ACO-E across different parameters and to compare against other similar algorithms. These algorithms were ACO-B (de Campos et al., 2002a), EPQ (Cotta and Muruzábal, 2004; Muruzábal and Cotta, 2004) and a greedy search in the space of equivalence classes using Chickering's operators (Chickering, 2002a) (called GREEDY-E here). A very brief description of these will now be given.

ACO-B ACO-E is based in part on the design of this algorithm and so there are some similarities. ACO-B is an ACO based algorithm that provides a search through the space of DAGs, with each of its moves being the addition of a directed arc to the current DAG. A more detailed description is given in Section 3.5.2.

EPQ This method uses an evolutionary programming algorithm that performs a search over the space of equivalence classes of DAGs. Like Chickering (2002a), they explicitly use CPDAGs (defined in Section 2.1.3) to represent the individuals, i.e. equivalence classes of DAGs. At each generation, from a population P , members of the population are selected using a binary tournament and mutated using the operators of Chickering. The best P out of $2P$ members (the original P plus the just-mutated P members) selected are then put forward into the next round, for T rounds.

GREEDY-E This algorithm uses the operators of Chickering to perform a greedy search in the space of CPDAGs. The results of tests performed by Chickering showed that the search generally performed better than search in the space of DAGs.

Parameter	Value
N'	4
κ	0.2
t_{max}	200
m	5, 7, 10, 12, 15, 20
ρ	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
q_0	0.7, 0.75, 0.8, 0.85, 0.9, 0.95
β	0.0, 0.5, 1.0, 1.5, 2.0, 2.5
t_{step}	201

Table 5.3: Parameter values for testing ACO-E with experimental condition 1

For the experiments reported in this thesis, testing involved the six standard networks presented in Section 5.1.1. Similar to Section 5.2.1 the BDeu scoring criterion was used. As suggested by Kayaalp and Cooper (2002) and by Heckerman et al. (1995), an equivalent sample size of 4 was used for the parameter priors. Also an empty structure prior with κ as defined by Heckerman et al. (1995) was used. For each individual run, 10,000 data were sampled from the network and used to construct the scoring function. Then for each combination of values for the parameter settings of ρ , q_0 , β and m , a run of the experiment was made for both the ACO-E and ACO-B algorithms. Different sets of random seeds were used for both algorithms. Whilst not optimal, the large number of experiments would negate the difference of individual results. The range of values that these parameters were taken from are shown in Table 5.3. Note that with a t_{step} value of 201, the local optimisation part of the algorithm will not be run.

In total this gave 1296 runs for each algorithm, for each network. As a consequence, this gave a total of 216 results for each setting of a parameter. In order to match this number of runs, the EPQ and GREEDY-E algorithm were also run 216 times each. It should be stressed that each run using a particular combination of parameters was done with a different data set sampled from the network. This technique guards against overfitting the parameters to a particular data set. As before, different data sets were sampled for different algorithms. Again, the large number of experiments would negate any differences due to different samplings.

5.3.1.2 Experimental Condition 2

Experimental condition 2 was designed to test ACO-E against other state-of-the-art Bayesian network structure learning algorithms. For these purposes the results found in the study conducted by Tsamardinos et al. (2006) was used. This study produced a thorough comparison of many different algorithms and made the results available, which allows the results for ACO-E to be compared against all of the algorithms used in the study. The various parameters used for ACO-E were kept as close as possible to those used by Tsamardinos et al.. The various algorithms that were compared against were: the max-min hill-climbing algorithm (MMHC) (Tsamardinos et al., 2006), the optimal reinsertion algorithm (OR) (Moore and Wong, 2003), the sparse candidate algorithm (SC) (Friedman et al., 1999c), a greedy search using the three standard operators as in Table 4.1 (GS), the PC algorithm (PC) (Spirtes et al., 2000), the three phase dependency analysis algorithm (TPDA) (Cheng et al., 2002) and the greedy equivalent search algorithm (GES) (Chickering, 2002b). A very brief description of these algorithms will now be given:

MMHC This algorithm is a hybrid based on the sparse candidate algorithm. It uses conditional independence testing as discussed in Section 2.4.7 to find good candidate parents and then performs a greedy search in the space of DAGs. This strategy limits the run time of the algorithm and makes it applicable to data sets with large numbers of variables.

OR This score-and-search algorithm in the space of DAGs utilises a single operator. This operator deletes all arcs incident to a node and then reinserts the best combination of arcs. It can be efficient on large data sets.

SC This hybrid algorithm was created for the purpose of examining data sets with large numbers of variables. It works by constraining the parent set a node can have in order to limit the amount of scores that need to be computed. This is done using a conditional independence based approach as discussed in Section 2.4.7. A score-and-search algorithm can then be applied that respects the constraints imposed in the first step.

GS This algorithm was one of the earliest used in the score-and-search paradigm of Bayesian network structure learning. It is very simple to implement, normally comprising the three operators of add, delete and reverse an arc. Scores can be computed locally, with the algorithm proceeding until no higher score can be found.

Parameter	N'	κ	t_{max}	m	ρ	q_0	β	t_{step}
Value	10	0.09	200	20	0.4	0.75	0.75	201

Table 5.4: Parameter values for testing ACO-E experimental condition 2

PC A conditional independence style algorithm, PC works by performing a series of χ^2 statistical tests on a data set in order to find CIs it needs. The amount of tests that need to be performed is constrained by the algorithm. Like all CI based approaches, it can suffer from a lack of data.

TPDA This conditional independence based algorithm works by using a polynomially bounded number of CI tests. However, it has been shown by Chickering and Meek (2006), that an assumption it makes (monotone DAG-faithfulness) is a bad one. Monotone DAG-faithfulness assumes that the more active paths between variables, the higher the mutual information. In the classes of DAGs it can learn, there are better algorithms that can learn in a smaller amount of time.

GES This score-and-search algorithm that works in the space of CPDAGs utilises two phases. In the first phase, arcs are added greedily starting from the empty graph using a given operator, until no improvement can be made. Then in the next phase, arcs are deleted using another operator, until no improvement can be made. This algorithm produces a perfect map of the probability distribution in the limit of a large sample size of the data.

For these experiments, testing involved four of the six standard networks presented in Section 5.1.1; Alarm, Barley, HailFinder and Mildew. These networks were used as the experiments of Tsamardinos et al. did not use the other two (Diabetes and Win95pts). For each run of the algorithm, 5000 data were generated by sampling the particular networks in question. This was chosen as opposed to the 10,000 data in Section 5.3.1.1, as this was the amount chosen by Tsamardinos et al.

As in condition 1, the BDeu scoring function was used. The value of both the equivalent sample size for this function and the ACO-E parameters are shown in Table 5.4. These values were chosen as they represented reasonable values that should perform well on most instances. The value of N' , κ and the prior function were identical to those used by Tsamardinos et al.. Each experiment was run 100 times for each network.

5.3.2 Evaluation Criteria

In the running of these experiments, various scoring metrics were picked to ascertain how well certain algorithms behaved. These were

- The scoring function used in running the experiments;
- The structural Hamming distance (SHD); and
- The equivalence-class structural Hamming distance (ESHD).

which are explained below.

5.3.2.1 The Scoring Function

For all experiments, the BDeu scoring function was used with differing parameters, depending on the experimental condition. Because these parameters were uniform given the condition, the score value of a Bayesian network structure can be used to compare the results of different algorithms. In terms of the BDeu score, this means that the higher the average score achieved, the better the results.

5.3.2.2 Structural Hamming Distance

In order to provide an objective measure of network structure reconstruction behaviour and to compare results against the work of Tsamardinos et al. (2006), the value of the structural Hamming distance (SHD) metric is given. This measures the difference between the learnt network and the gold-standard generating network. The definition of this measure as given by Tsamardinos et al. is shown in Algorithm 5.2. Both networks are transformed from DAG to CPDAG if not already in this representation and a uniform penalty is given for each missing and extra edge and for each incorrectly directed arc. The SHD is a useful measure for comparing structural dissimilarity. However, upon examination, it can be noticed that this metric over penalises certain structures. In particular, edges in a graph that are compelled but do not participate in a v-structure can become reversible by the removal of the v-structures that ultimately compel those edges and vice-versa. From this, a single change in a graph can have cascading effects that cause a large subset of arcs to become compelled or reversible. Because the SHD does not take into account whether a compelled arc participates in a v-structure or not, it can over differentiate between two graphs. To stop this happening a metric is needed that can take the above into account.

Algorithm 5.2 The SHD metric

Input: PDAG \mathcal{H} , \mathcal{G} **Output:** shd

```

 $shd \leftarrow 0$ 
for every edge  $E$  different in  $\mathcal{H}$  than  $\mathcal{G}$  do
  if  $E$  is missing in  $\mathcal{H}$  then
     $shd \leftarrow shd + 1$ 
  end if
  if  $E$  is extra in  $\mathcal{H}$  then
     $shd \leftarrow shd + 1$ 
  end if
  if  $E$  is incorrectly oriented in  $\mathcal{H}$  then
    {i.e. if  $E$  is reversed or the directedness is different}
     $shd \leftarrow shd + 1$ 
  end if
end for
return  $shd$ 

```

5.3.2.3 The Equivalence-Class Structural Hamming Distance

According to Verma and Pearl (1991), two PDAGs representing equivalence classes are equal if they contain the same skeleton and same set of v-structures. Looking at this, the task of finding the distance between two PDAGs can be broken down into finding the distance between the skeletons of the PDAGs and the set of v-structures of the PDAGs. The difference between the skeletons can be given two quantities – the number of arcs that need to be added and the number of arcs that need to be deleted in order to change one PDAG skeleton to the other. Finding the difference between the two sets of v-structures could be given in a similar fashion. However, directing an arc, or adding a directed arc from x to y results in an increase to the number of v-structures by the number of directed arcs incident to y . I.e. a single change to an arc can result in a large change to the number of v-structures. An example of this is shown in Figure 5.7. Here, when a single arc is added from y to x , 3 extra v-structures are added.

To counter this, instead of counting differences between v-structures, the number of arc directings (i.e. transforming an undirected arc into a directed arc), directed arc additions and or arc reversals between the PDAGs that produce at least one new v-structure are counted. The same is done for arc indirectings (i.e. transforming a directed

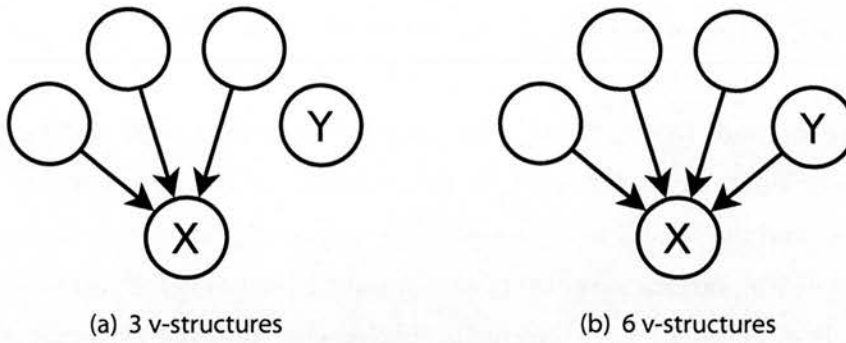


Figure 5.7: Adding v-structures

arc into an undirected one), directed arc deletions and arc reversals between the PDAGs that remove at least one v-structure. Both of these procedures give two quantities that give a measure related to the number of v-structures added and deleted between two PDAGs. In the case of Figure 5.7, the distance between the two graphs is 1 for the skeleton arc added and 1 for the v-structure compulsion.

Given these four quantities (Skeleton arcs added, Skeleton arcs removed, V-structure compulsions, V-structure removals), a single measure that shows the distance between two PDAGs can be obtained. An example of a simple measure that will be used in this work is obtained by adding all the quantities together. This will be defined as the Equivalence-Class Structural Hamming Distance (ESHD). Whilst similar to the SHD metric, it is intended to better capture differences between PDAGs, by taking into account cascading changes. As such, it is a better representation of a Hamming distance between two PDAGs, in that it measures how many basic edits are needed to change one PDAG to another.

5.4 Summary

In this chapter, the methodology used for conducting experiments into the effectiveness of the techniques of Chapter 4 was given. This was presented in three parts:

- an explanation of the standard Bayesian networks used for testing the methods;
- the design and evaluation criteria for testing the methods to speed up the learning process; and
- the design and evaluation criteria for testing the effectiveness of the ACO-E algorithm.

For the standard networks, six well-known Bayesian networks were described, with pointers to their structure and a list of some of their properties.

With the methods to speed up the learning process, the design of the experiments involved the amount of experiments to be run, the amount of data to sample, the scoring criteria to use and the selection of the algorithm they would be used with. For this cache updating algorithm, various parameters were specified, including the various operators to be used. Evaluation criteria were selected to specify what information would be recorded.

For the experiments involving ACO-E, their design involved two conditions. Condition 1 involved testing ACO-E with parameters being set to various values. This would enable the behaviour of the algorithm to be better understood and allow testing against other metaheuristic algorithms that are similar. Condition 2 involved testing ACO-E with parameters being set to reasonable values that should produce good results in most circumstances. This would enable testing the algorithm to be tested against other state-of-the-art Bayesian network structure learning methods.

The next chapter will present the results of the experiments run with these conditions and also give a discussion of, and interpretation to, the meaning of the results.

Chapter 6

Results and Discussion

THIS CHAPTER will present the results arising from experiments conducted according to the methodology in Chapter 5. Following this will be a discussion into the behaviour of the various methods of Chapter 4, depending on any parameters and other conditions that might influence their behaviour.

In order to investigate the methods of Section 4.2 that are designed to speed up the running time of structure learning algorithms, this discussion will look at the behaviour of these methods as a function of the number of variables in the data set.

For the ACO-E algorithm, the discussion will focus on different aspects:

- The behavior of ACO-E as a function of its main parameters: heuristic power, pheromone deposition and exploration/exploitation tradeoff;
- The behaviour of ACO-E compared to other metaheuristic algorithms, both as a function of time and final result behaviour; and
- The behaviour of ACO-E compared to state-of-the-art algorithms for Bayesian network structure learning, as a function of reconstructive ability.

6.1 Accelerating the Learning Process

In this section, the results from the experiments described in Section 5.2 will be presented, followed by a discussion of the effectiveness of the methods as described in Section 4.2.

	Original	New	Original/New	$ V $
Alarm ($10^{-3}s$)	3.5603 ± 0.0390	0.9753 ± 0.1363	3.6505	37
Barley ($10^{-3}s$)	8.5345 ± 0.0465	1.7223 ± 0.2844	4.9553	48
Diabetes ($10^{-3}s$)	3.8889 ± 0.0666	1.0241 ± 0.2093	3.7974	36
HailFinder ($10^{-4}s$)	1.1404 ± 0.1512	0.1431 ± 0.0260	7.9693	56
Mildew ($10^{-3}s$)	2.3267 ± 0.3299	0.6540 ± 0.0394	3.5576	35
Win95pts ($10^{-4}s$)	7.3302 ± 1.2948	0.6527 ± 0.1014	11.2306	76

Table 6.1: Mean and standard deviation of running times at $t = 100$ over 100 runs

6.1.1 Results

The first results illustrated are those as a function of time and are presented in Figures 6.1 to 6.6. The analysis of these results occurs in the next section. Results for each of the two conditions (using the methods described in Section 4.2 and not using these methods) are indicated on each of the graphs. Here, the run time is given for an experiment at each iteration of the ACO-E algorithm, averaged across the 100 experiment runs. Note that the graphs are designed to show the difference between the two experimental conditions. As such the absolute scale on each graph varies according to the example network. To quantify these results further, the original running times and improved running times for each of the test networks at iteration $t = 100$ are shown in Table 6.1. Also given are the ratio of original to improved running times and the number of nodes $|V|$ in each network. Finally, Figure 6.7 gives the average runtimes over all datasets presented in Table 6.1 as a function of the number of nodes in the network and Figure 6.8 shows the speed-up ratio as a function of the number of nodes in the network.

6.1.2 Discussion

The results given in Figures 6.1 to 6.6 show that there is a speed-up effect across the data sets. In all the examples, the methods from Section 4.2 make the ACO-E algorithm faster than when the methods are not employed. This is most evident at the ends of the runs, i.e. at ant iteration 100.

It also appears that for each network, this speed-up is linear with respect to the time the algorithm is run. In other words, the ratio of times is constant at any given point in the algorithm, for a given network. This is because the rate of both of the conditions appears constant, so therefore their ratio would also be constant.

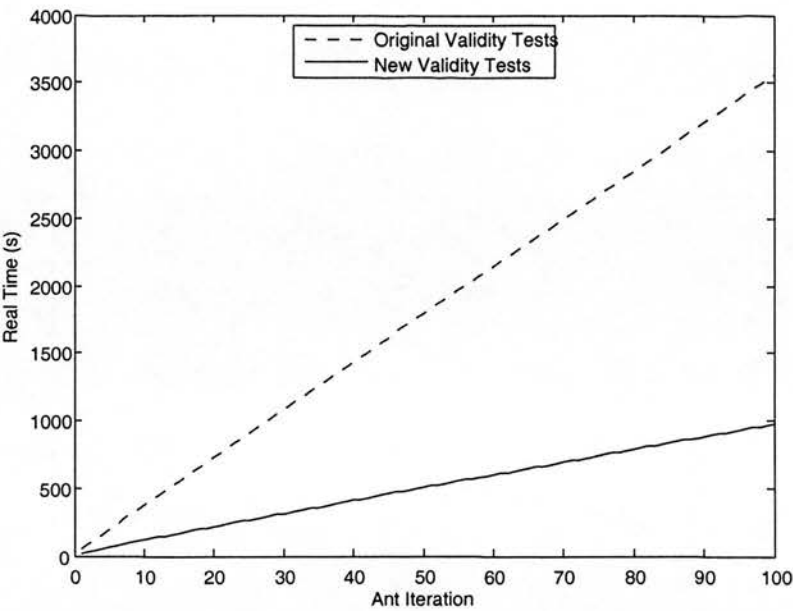


Figure 6.1: Comparison of original and new validity checking – Alarm

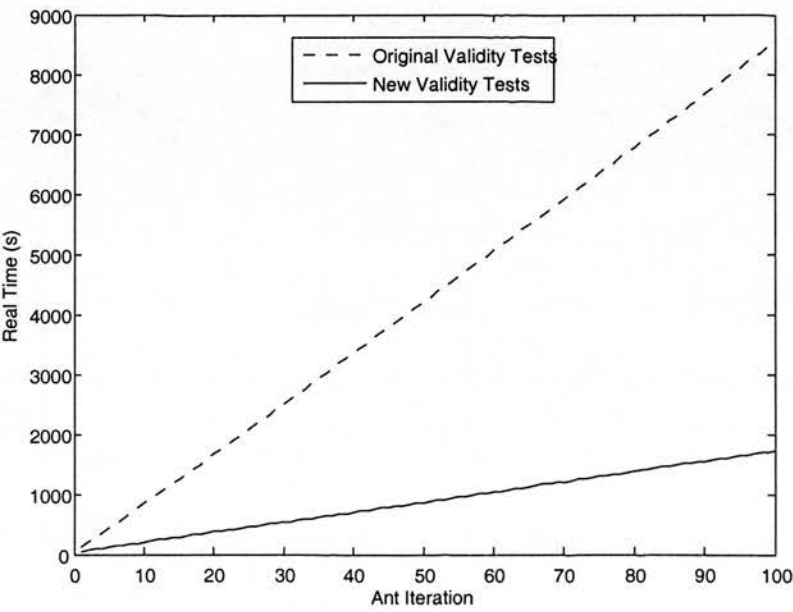


Figure 6.2: Comparison of original and new validity checking – Barley

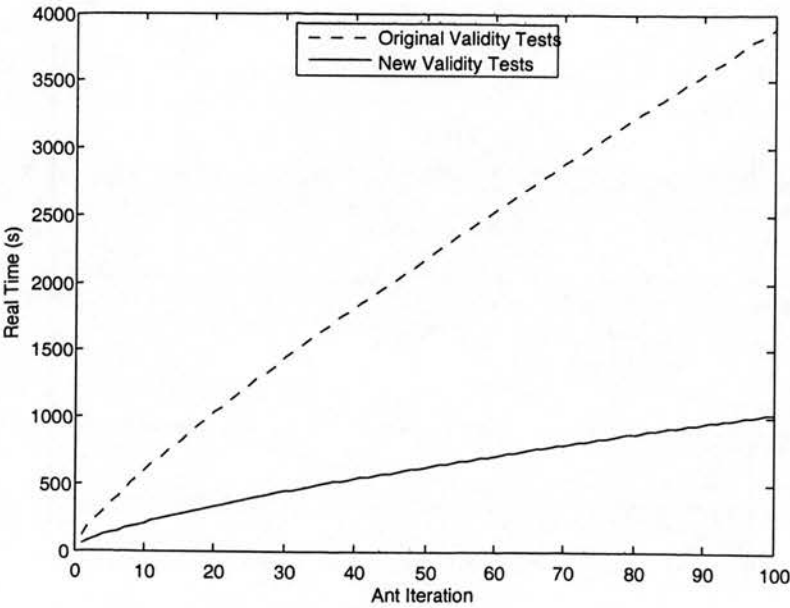


Figure 6.3: Comparison of original and new validity checking – Diabetes

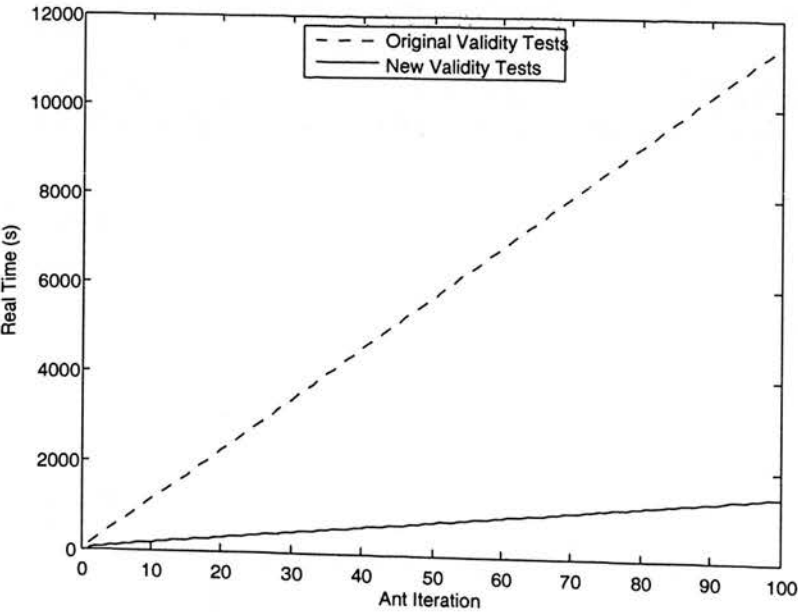


Figure 6.4: Comparison of original and new validity checking – HailFinder

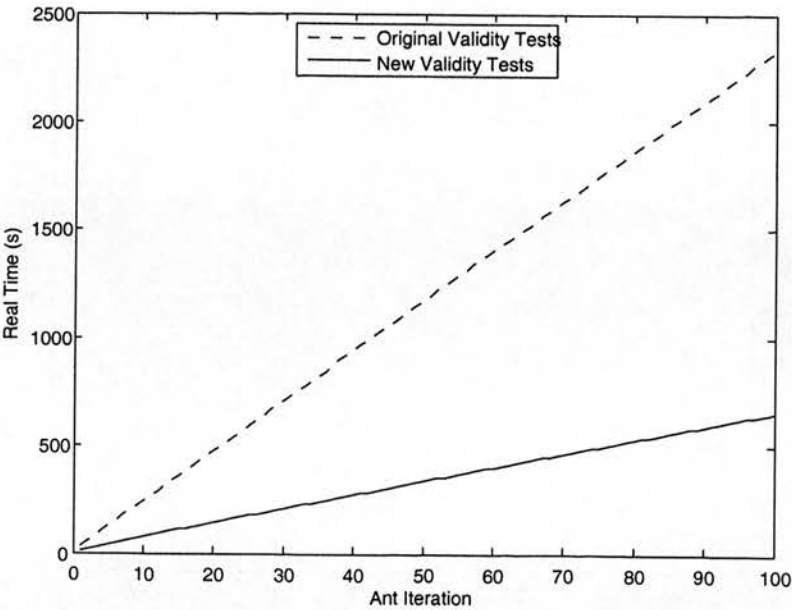


Figure 6.5: Comparison of original and new validity checking – Mildew

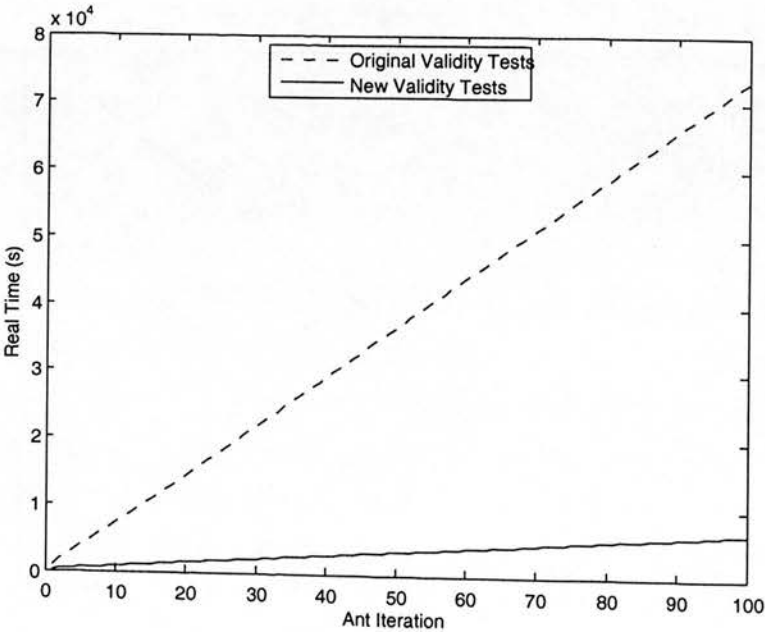


Figure 6.6: Comparison of original and new validity checking – Win95pts

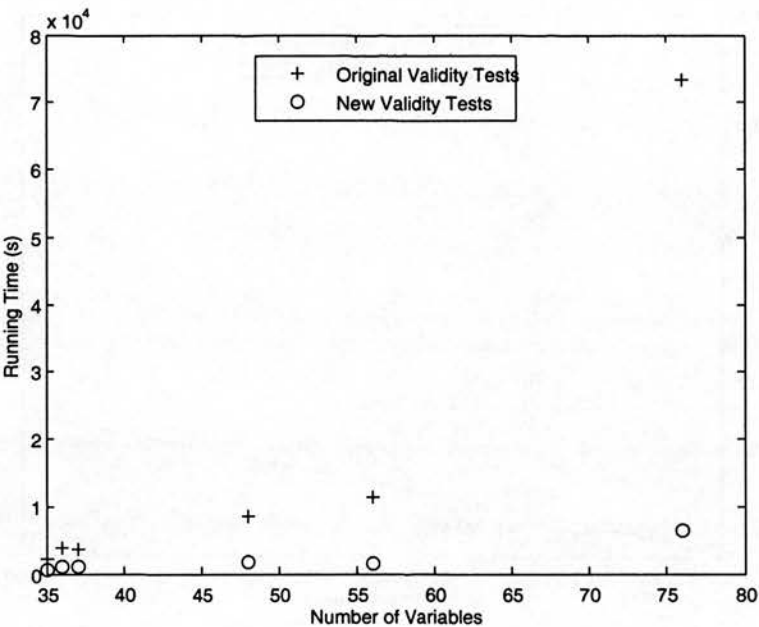


Figure 6.7: Original and new running times

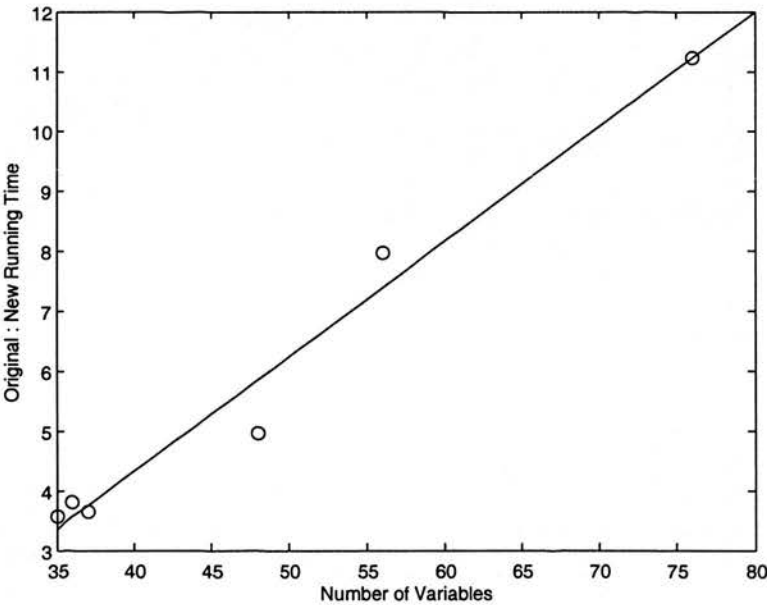


Figure 6.8: Speed-up ratio

Finally, looking at Figures 6.7 and 6.8 it appears that the ratio of times is linear in the number of variables in the test network. If this is true and if it turns out that the running time using the speeded up methods is linear in the number of variables, then the running time without the speeded up methods would be quadratic in the number of variables. Another way of saying this would be that the running time using the speeded up methods would be a function of the square-root of the running time without using these methods.

In the next sections, these ideas will be tested statistically in order to provide a more quantitative answer as to their validity. This will also provide confirmation of the analytical results obtained in Section 4.2.1. In order that this can be done, certain standard statistical tests will be used.

6.1.2.1 Pearson's Correlation Coefficient

The first of these is the product-moment correlation coefficient as given by Pearson (1896). This statistic is a measure of the linear relationship between two random variables and is defined by

$$r = \frac{\text{cov}(X, Y)}{s_X s_Y},$$

where s_X and s_Y are the sample standard deviations of X and Y and $\text{cov}(X, Y)$ is the covariance of X and Y , given by

$$\text{cov}(X, Y) = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y}),$$

where \bar{X} and \bar{Y} are the sample arithmetic means of X and Y respectively. r can range between the values -1 and 1 :

- At $r = 1$ there is a definite positive linear relationship between the variables;
- At $r = -1$ there is a definite negative linear relationship between the variables; and
- At $r = 0$ there is no linear relationship between the variables.

In calculating whether a particular value of r indicates that a null hypothesis can be rejected, critical values of r can be looked up in tables. These critical values indicated the boundary between which a statistic will confirm or reject the null hypothesis. If $|r|$ as calculated is greater than the critical value used, then the null hypothesis can be rejected. To find the needed critical value, two pieces of information are needed: the degrees of freedom of the sample ν and the significance level α at which the test will be made. α

indicates the probability that an error will be made in rejecting the null hypothesis. With two variables, ν for a sample is equal to $n - 2$, where n is the size of the sample.

6.1.2.2 Welch's t -test

Another statistical test which will be used is the t -test proposed by Welch (1947). As opposed to the more widely known t -test by Student (1908), Welch's test can operate on samples with unequal variances. It is generally used to compare the means of two different samples. The statistic t is calculated as

$$t = \frac{\bar{X} - \bar{Y}}{\sqrt{\frac{s_X^2}{n_X} + \frac{s_Y^2}{n_Y}}},$$

where \bar{X} and \bar{Y} are the sample arithmetic means of X and Y , s_X^2 and s_Y^2 are the sample variances of X and Y and n_X and n_Y are the sample sizes for X and Y . Similar to Pearson's correlation coefficient, to calculate whether a significant difference has been found, a critical value must be looked up. If the calculated t is greater than this critical value, then the null hypothesis can be rejected. Also similar to Pearson's correlation coefficient, to find this critical value, two other pieces of information are needed, the degrees of freedom of the sample ν and the significance level α . To calculate ν , the Welch-Satterthwaite approximation (Satterthwaite, 1946) is normally used

$$\nu = \frac{\left(\frac{s_X^2}{n_X} + \frac{s_Y^2}{n_Y} \right)^2}{\frac{s_X^4}{n_X^2 \nu_X} + \frac{s_Y^4}{n_Y^2 \nu_Y}}.$$

If the value of ν is much over 120, then the distribution of t is roughly normal, so a check can be made against a table of normal values.

6.1.2.3 Testing Whether Run-times are Significantly Different

The first test to be conducted will be comparing the run times of each condition of a given network. These results are given in Table 6.1 and the tests will check whether the difference between the means are significant. To do this, Welch's t -test as shown in Section 6.1.2.2 will be used. In this case, the test will be two-tailed. Table 6.2 shows the results from performing this test across the six networks. As can be seen from the p-value column, there is very strong evidence to reject the null hypothesis, i.e. it is very likely that the means are different. Therefore, the methods of Section 4.2 are effective in speeding up the running times of the ACO-E algorithm. The next sections will seek to

	t	ν	p-value
Alarm	62.56	122.85	5.0×10^{-95}
Barley	124.76	164.20	1.3×10^{-164}
Diabetes	41.05	118.37	7.5×10^{-72}
HailFinder	64.99	104.85	1.6×10^{-86}
Mildew	50.35	101.82	9.1×10^{-74}
Win95pts	51.41	100.22	7.7×10^{-74}

Table 6.2: t values and p-values for comparing the average run times

determine the nature of this speed-up, i.e. is it possible to determine *how* much faster this speed-up will be?

6.1.2.4 Testing Whether the Speed-up Ratio is Constant Across Iterations

Looking at Figures 6.1 to 6.6, it appears that the rate at which the algorithm's runtime increases is constant, both for the original validity tests and the new validity tests, i.e.

$$\frac{O}{I} = c \quad \frac{F}{I} = d,$$

where O is the value of the run time up to the current iteration using the original validity tests, F is the value of the run time up to the current iteration using the new validity tests and I is the number of iterations.

If this is the case, then the speed-up ratio is also constant

$$\frac{O}{F} = \frac{\frac{O}{I}}{\frac{F}{I}} = \frac{c}{d}.$$

In order to conclude that the rates are constant at each iteration, a test using Pearson's correlation coefficient can be used. Table 6.3 shows r values and p-values for each of the six networks using both the original and faster validity tests. These r values are obtained from the mean of the time at each point t from 1 to 100. In a sense, they are measuring the linearity of the slope on each of the graphs in Figures 6.1 to 6.6. It can be seen that the r values are consistently high and the p-values consistently low; in all cases there is very strong evidence that the relationships are linear. Because of this, it can be said that the speed-up ratio is constant over all iterations. In other words, the speed-up ratio is invariant to the number of iterations.

	Original		New	
	r	p-value	r	p-value
Alarm	1.0000	3.3×10^{-307}	0.9999	5.2×10^{-190}
Barley	1.0000	2.0×10^{-198}	1.0000	2.2×10^{-202}
Diabetes	0.9986	2.7×10^{-127}	0.9949	2.5×10^{-99}
HailFinder	1.0000	1.0×10^{-229}	1.0000	5.1×10^{-213}
Mildew	1.0000	3.1×10^{-240}	0.9999	2.1×10^{-190}
Win95pts	1.0000	1.8×10^{-239}	1.0000	1.8×10^{-223}

Table 6.3: r values and p-values for comparing the average run times across iterations

6.1.2.5 Testing Speed-up Ratio as a Function of the Number of Variables

Looking at Figure 6.8 it seems that there might be a linear relationship between the speed-up ratio and the number of variables in the network, i.e. the amount of speed-up depends on the number of variables. To test this, the Pearson correlation coefficient was calculated over the ratio and number of nodes and came to the figure $r = 0.9870$ and a p-value of 0.00025. The critical value of r for $\nu = 4$ degrees of freedom and $\alpha = 0.01$ was found to be $r = 0.917$. Therefore there is a very high probability that a linear relationship exists between the number of variables in a model and the speed-up ratio given by the methods introduced in this paper. Put another way

$$\frac{O}{F} = kN, \quad (6.1)$$

where k is a constant and N is the number of variables in a network.

Looking again at Figure 6.7, there appears to be a linear relationship between the number of variables in a network and the runtime with the new acceleration techniques given that network. This was tested by calculating the Pearson correlation coefficient over these values. This came to a figure of $r = 0.9144$ and a p-value of 0.0107. Whilst not definitive, these results are evidence that there is a linear relationship between the number of variables in a network and the faster running time, i.e. that

$$F = lN.$$

From Equation 6.1, this implies that

$$O = lkN^2$$

i.e. that the original running time is quadratic in the number of variables.

6.1.2.6 A Comparison of the Results to the Theory

Recalling the content of Section 4.2.1, it was shown that the number of validity tests that needed to be performed was reduced from being in $O(n^2)$ to $O(nk^2)$, where n is the number of nodes and k is a constant which bounds the number of parents, children and neighbours a node can have. This reduction in complexity from quadratic behaviour to linear reflects that of the empirical results obtained, where there is evidence that a similar reduction has taken place. From these two pieces of information, there are strong grounds to believe that using the faster validity testing scheme speeds up the run time of ACO-E, a heuristic search algorithm that comprises multiple restarts.

Quantifying the behaviour of the caching system is more problematic, as there exists no theory as to how it might work. Generally, by looking at a typical trace of an algorithm it could be seen how the cache might work, in this case for the InsertU operator:

- Initially the graph is sparsely connected by undirected links. This will mean that fewer cache entries need to be updated.
- Eventually the graph becomes densely connected by undirected links. This means at each stage, each entry in the cache must be updated.
- At some stage, the MakeV operator directs two edges. This causes a cascade of edges to become directed.
- This means that the graph is disconnected by undirected links and so returns to the early behaviour of having fewer InsertU cache entries updated. At this point, cache entries for InsertD and ReverseD will be prevalent.

Each operator might have its own typical behaviour, which works differently depending on the connectivity of the graph. Because of this, analysis of the computational complexity is difficult. However, looking at how the algorithm behaves as above, the cache performs better with a sparse graph. With an almost empty graph, the cache needs to update very few entries, and so the validity checking procedure would finish in a small constant time, irrespective of the number of nodes in the graph. In this case, the speed-up is in the order of $O(n^2)$.

With a graph that is almost completely connected, many entries would have to be updated. In this case, there would be no benefit to the cache. The applicability of this method would, therefore, depend on how highly connected the final graph would be. With a high degree of connectivity, any benefits of using a cache would be confined to

the start of a run. With a lower degree of connectivity, these would be felt throughout the run.

6.2 Using Ant Colony Optimisation in Learning an Equivalence Class

In this section the results of experiments performed according to the methodologies given in Section 5.3 will be presented. In 5.3, two experimental conditions were given. The first dealt with analysing the behaviour of ACO-E with respect to its parameters and in comparison to other metaheuristic algorithms that shared similar behaviour. The second condition dealt with comparing ACO-E to other state-of-the-art Bayesian network structure learning algorithms. These results will be presented in this order, followed by a discussion and interpretation of these results.

6.2.1 Results

The results for experimental condition 1 and condition 2 are presented below.

6.2.1.1 Experimental Condition 1

The results of the runs using experimental condition 1 are shown in two sets, which reflect how they will be analysed later. Firstly, detailed results for ACO-E are shown in Tables 6.4, 6.5, 6.6 and 6.7. In these tables, the figures given for each parameter value are the results averaged over all other parameters; e.g. the figure for $\rho = 0.1$ is given by calculating the mean and standard deviation over all results with $\rho = 0.1$. In this case, the size of the samples will be 216, and will be calculated over all combinations of the values of the other parameters m , ρ , q_0 and β as shown in Table 5.3 and using the values of N' , κ , t_{max} and t_{step} specified in that table. It should be noted that the specific values of $\rho = 0$ and $\beta = 0$ are special cases. When $\rho = 0$, there is no pheromone evaporation and no pheromone deposition on the graph; i.e. pheromone plays no part in the algorithm. With $\beta = 0$, there is no heuristic used whilst the ants traverse the construction graph.

The same results are presented differently in Table 6.8 and Figures 6.9 to 6.26. These show the behaviour of ACO-E against other algorithms, both as a function of the algorithm iteration (Figures 6.9 to 6.26) and as a final value (Table 6.8). In these results, the iterations figure is that for ACO-E and ACO-B. The EPQ iteration number is three times that of the shown iteration. As such, whilst ACO-E and ACO-B were run for 200

iterations, EPQ was run for 600 and the results scaled to 200. This was done so as to compare the steady state behaviour of the algorithms when they were close to their limiting values. As such, the number of iterations is somewhat arbitrary.

	ρ						
	0.0	0.1	0.2	0.3	0.4	0.5	
Alarm ($\times -10^5$)	1.0383 \pm 0.0037	1.0385 \pm 0.0036	1.0387 \pm 0.0035	1.0385 \pm 0.0037	1.0388 \pm 0.0037	1.0380 \pm 0.0038	
Barley ($\times -10^5$)	5.0756 \pm 0.0136	5.0697 \pm 0.0039	5.0702 \pm 0.0096	5.0696 \pm 0.0039	5.0699 \pm 0.0041	5.0699 \pm 0.0041	
Diabetes ($\times -10^5$)	1.9394 \pm 0.0032	1.9391 \pm 0.0034	1.9394 \pm 0.0029	1.9386 \pm 0.0034	1.9395 \pm 0.0034	1.9393 \pm 0.0035	
HailFinder ($\times -10^5$)	4.9207 \pm 0.0039	4.9206 \pm 0.0038	4.9204 \pm 0.0042	4.9210 \pm 0.0034	4.9202 \pm 0.0040	4.9207 \pm 0.0037	
Mildew ($\times -10^5$)	4.5426 \pm 0.0096	4.5412 \pm 0.0091	4.5417 \pm 0.0101	4.5401 \pm 0.0094	4.5388 \pm 0.0083	4.5395 \pm 0.0090	
Win95pts ($\times -10^4$)	9.4322 \pm 0.0448	9.4169 \pm 0.0433	9.4086 \pm 0.0452	9.4125 \pm 0.0454	9.4154 \pm 0.0457	9.4210 \pm 0.0468	
	q_0						
	0.7	0.75	0.8	0.85	0.9	0.95	
Alarm ($\times -10^5$)	1.0385 \pm 0.0035	1.0388 \pm 0.0035	1.0383 \pm 0.0035	1.0382 \pm 0.0035	1.0386 \pm 0.0039	1.0383 \pm 0.0040	
Barley ($\times -10^5$)	5.0703 \pm 0.0066	5.0704 \pm 0.0065	5.0705 \pm 0.0060	5.0708 \pm 0.0057	5.0710 \pm 0.0073	5.0720 \pm 0.0126	
Diabetes ($\times -10^5$)	1.9391 \pm 0.0035	1.9391 \pm 0.0033	1.9393 \pm 0.0032	1.9393 \pm 0.0035	1.9393 \pm 0.0035	1.9390 \pm 0.0030	
HailFinder ($\times -10^5$)	4.9207 \pm 0.0039	4.9208 \pm 0.0035	4.9208 \pm 0.0038	4.9205 \pm 0.0038	4.9204 \pm 0.0040	4.9204 \pm 0.0040	
Mildew ($\times -10^5$)	4.5362 \pm 0.0069	4.5378 \pm 0.0074	4.5389 \pm 0.0084	4.5410 \pm 0.0098	4.5430 \pm 0.0097	4.5472 \pm 0.0092	
Win95pts ($\times -10^4$)	9.4200 \pm 0.0453	9.4183 \pm 0.0441	9.4199 \pm 0.0462	9.4192 \pm 0.0481	9.4160 \pm 0.0468	9.4130 \pm 0.0439	

Table 6.4: Mean and standard deviation of the BDeu score for ACO-E for each parameter setting (the bold figure is the best for that row)

	β							
	0.0	0.5	1.0	1.5	2.0	2.5		
Alarm ($\times -10^5$)	1.0387 \pm 0.0036	1.0378 \pm 0.0038	1.0383 \pm 0.0036	1.0386 \pm 0.0035	1.0387 \pm 0.0040	1.0387 \pm 0.0034		
Barley ($\times -10^5$)	5.0775 \pm 0.0121	5.0694 \pm 0.0043	5.0689 \pm 0.0035	5.0696 \pm 0.0040	5.0697 \pm 0.0055	5.0698 \pm 0.0097		
Diabetes ($\times -10^5$)	1.9389 \pm 0.0033	1.9390 \pm 0.0034	1.9394 \pm 0.0034	1.9393 \pm 0.0032	1.9394 \pm 0.0032	1.9391 \pm 0.0035		
HailFinder ($\times -10^5$)	4.9212 \pm 0.0039	4.9205 \pm 0.0037	4.9203 \pm 0.0040	4.9206 \pm 0.0038	4.9205 \pm 0.0038	4.9205 \pm 0.0040		
Mildew ($\times -10^5$)	4.5378 \pm 0.0075	4.5371 \pm 0.0075	4.5392 \pm 0.0090	4.5404 \pm 0.0094	4.5440 \pm 0.0102	4.5456 \pm 0.0090		
Win95pts ($\times -10^4$)	9.4515 \pm 0.0505	9.4092 \pm 0.0404	9.4135 \pm 0.0420	9.4101 \pm 0.0385	9.4116 \pm 0.0444	9.4106 \pm 0.0427		
<hr/>								
	m							
	<hr/>							
	5	7	10	12	15	20		
Alarm ($\times -10^5$)	1.0383 \pm 0.0036	1.0386 \pm 0.0036	1.0381 \pm 0.0037	1.0389 \pm 0.0037	1.0384 \pm 0.0039	1.0384 \pm 0.0036		
Barley ($\times -10^5$)	5.0721 \pm 0.0120	5.0707 \pm 0.00061	5.0709 \pm 0.0083	5.0704 \pm 0.0066	5.0704 \pm 0.0061	5.0705 \pm 0.0060		
Diabetes ($\times -10^5$)	1.9392 \pm 0.0031	1.9392 \pm 0.0036	1.9391 \pm 0.0032	1.9392 \pm 0.0032	1.9391 \pm 0.0033	1.9394 \pm 0.0034		
HailFinder ($\times -10^5$)	4.9202 \pm 0.0041	4.9209 \pm 0.0037	4.9201 \pm 0.0039	4.9210 \pm 0.0040	4.9204 \pm 0.0036	4.9210 \pm 0.0037		
Mildew ($\times -10^5$)	4.5440 \pm 0.0097	4.5427 \pm 0.0097	4.5409 \pm 0.0093	4.5392 \pm 0.0092	4.5386 \pm 0.0082	4.5385 \pm 0.0085		
Win95pts ($\times -10^4$)	9.4201 \pm 0.0452	9.4190 \pm 0.0452	9.4178 \pm 0.0457	9.4188 \pm 0.0458	9.4164 \pm 0.0462	9.4145 \pm 0.0467		

Table 6.5: Mean and standard deviation of the BDeu score for ACO-E for each parameter setting (the bold figure is the best for that row)

ρ						
	0.0	0.1	0.2	0.3	0.4	0.5
Alarm	6.9 ± 4.9	5.6 ± 3.1	6.0 ± 3.2	5.6 ± 3.3	5.9 ± 3.0	5.5 ± 3.0
Barley	56.4 ± 10.8	52.8 ± 3.9	53.0 ± 4.1	53.2 ± 4.4	52.6 ± 4.0	52.9 ± 4.8
Diabetes	63.5 ± 5.8	65.5 ± 5.3	65.0 ± 5.4	65.1 ± 5.6	64.2 ± 5.4	63.7 ± 4.8
HailFinder	50.6 ± 6.9	50.5 ± 6.8	51.0 ± 7.8	51.8 ± 7.8	51.8 ± 6.7	51.1 ± 7.9
Mildew	25.7 ± 5.8	22.6 ± 5.0	22.8 ± 5.1	22.0 ± 4.9	21.4 ± 4.7	21.9 ± 4.8
Win95pts	94.6 ± 27.7	83.3 ± 24.3	81.7 ± 20.3	81.9 ± 22.9	81.9 ± 24.2	83.5 ± 21.9
q_0						
	0.7	0.75	0.8	0.85	0.9	0.95
Alarm	5.2 ± 2.5	5.6 ± 3.1	5.4 ± 3.0	6.1 ± 3.7	6.4 ± 4.2	6.7 ± 4.1
Barley	53.9 ± 5.9	53.6 ± 5.8	53.3 ± 5.4	53.4 ± 5.9	53.1 ± 5.8	53.6 ± 7.2
Diabetes	62.1 ± 4.8	62.6 ± 4.9	63.4 ± 4.9	64.7 ± 5.4	66.1 ± 5.3	68.1 ± 4.7
HailFinder	52.1 ± 7.3	51.9 ± 7.5	51.5 ± 8.2	50.7 ± 7.1	50.6 ± 7.7	50.0 ± 5.9
Mildew	20.2 ± 3.8	21.1 ± 4.7	21.5 ± 4.8	22.6 ± 5.1	24.3 ± 5.4	26.6 ± 4.9
Win95pts	90.3 ± 23.5	88.1 ± 24.5	84.9 ± 22.6	83.2 ± 22.5	81.0 ± 22.2	79.3 ± 27.1
β						
	0.0	0.5	1.0	1.5	2.0	2.5
Alarm	7.4 ± 5.1	4.3 ± 1.1	4.9 ± 2.4	5.4 ± 2.8	6.2 ± 3.6	7.2 ± 3.7
Barley	61.4 ± 9.0	52.0 ± 3.9	51.9 ± 3.0	51.7 ± 3.3	51.9 ± 3.3	52.0 ± 3.7
Diabetes	64.3 ± 4.9	64.3 ± 5.5	64.2 ± 5.5	64.7 ± 5.1	64.6 ± 5.9	64.8 ± 5.5
HailFinder	52.2 ± 7.0	52.0 ± 6.5	51.5 ± 6.7	50.1 ± 8.0	50.3 ± 8.0	50.5 ± 7.5
Mildew	20.9 ± 4.9	20.9 ± 4.0	21.9 ± 5.0	22.4 ± 5.1	24.6 ± 5.4	25.7 ± 5.2
Win95pts	109.1 ± 28.6	89.6 ± 23.9	79.8 ± 17.9	77.0 ± 17.2	77.1 ± 19.3	74.4 ± 15.5
m						
	5	7	10	12	15	20
Alarm	7.1 ± 4.3	6.6 ± 3.8	5.9 ± 3.5	5.2 ± 2.8	5.7 ± 3.6	5.1 ± 2.4
Barley	54.3 ± 7.3	52.8 ± 5.3	53.8 ± 6.6	53.2 ± 5.7	53.7 ± 5.8	53.1 ± 5.1
Diabetes	66.2 ± 5.1	65.8 ± 5.9	64.1 ± 5.4	64.5 ± 5.6	63.5 ± 5.4	62.8 ± 4.7
HailFinder	50.5 ± 7.2	50.5 ± 6.7	51.0 ± 6.5	51.8 ± 7.0	51.4 ± 8.5	51.6 ± 7.9
Mildew	24.6 ± 5.3	23.9 ± 5.6	22.7 ± 5.5	22.2 ± 4.9	21.6 ± 4.7	21.4 ± 4.7
Win95pts	83.3 ± 23.3	83.0 ± 21.9	85.6 ± 26.2	85.5 ± 22.1	85.5 ± 25.0	84.0 ± 25.5

Table 6.6: Mean and standard deviation of the SHD for ACO-E for each parameter setting (the bold figure is the best for that row)

ρ						
	0.0	0.1	0.2	0.3	0.4	0.5
Alarm	4.6±3.0	3.8±1.7	4.0±1.6	3.8±1.8	3.9±1.6	3.7±1.6
Barley	41.1±7.8	38.4±3.2	38.8±3.5	38.8±3.2	38.2±3.2	38.2±3.4
Diabetes	43.2±3.7	44.4±3.4	44.2±3.6	44.0±3.6	43.5±3.6	43.1±3.2
HailFinder	30.3±4.7	30.0±4.9	30.5±5.2	31.1±5.8	31.0±4.8	31.2±5.1
Mildew	19.5±4.3	17.3±4.1	17.4±4.1	16.7±4.0	16.3±3.8	16.7±3.9
Win95pts	78.2±22.2	68.3±19.6	67.1± 16.6	67.4±18.1	66.9±19.8	69.1±18.0
q_0						
	0.7	0.75	0.8	0.85	0.9	0.95
Alarm	3.6±1.4	3.8±1.7	3.7±1.7	4.1±2.1	4.3±2.4	4.4±2.3
Barley	39.2±4.5	38.9±4.1	38.8± 4.0	38.8±4.4	38.7±4.2	39.1±5.4
Diabetes	42.3±3.2	42.5±3.2	43.0±3.2	43.9±3.6	44.7±3.4	46.2±3.0
HailFinder	31.3±5.6	31.3±5.6	31.3±5.8	30.7±4.8	30.0±4.9	29.5±3.1
Mildew	15.3±3.0	15.9±3.5	16.3±3.8	17.3±4.1	18.5±4.3	20.4±3.9
Win95pts	74.1±18.9	72.4±19.9	70.1±18.1	68.7±18.6	66.8±17.9	65.1±22.1
β						
	0.0	0.5	1.0	1.5	2.0	2.5
Alarm	5.1±3.2	3.2±0.6	3.4±1.2	3.7±1.4	4.0±1.8	4.9±1.9
Barley	43.8±6.5	38.0±3.5	38.0± 2.9	37.7±3.2	37.9±3.0	38.1±3.4
Diabetes	43.7±3.2	43.8±3.8	43.6±3.6	43.8±3.5	43.7±3.8	43.8±3.4
HailFinder	31.3±5.2	30.9± 5.0	30.9±5.2	30.3±5.1	30.6± 5.0	30.2±5.0
Mildew	15.8±3.6	15.7±3.2	16.6±4.0	17.1±4.1	18.8±4.3	19.7±4.2
Win95pts	90.1±22.8	73.1±19.5	66.0±14.3	63.1±13.9	63.8±15.5	60.9±12.4
m						
	5	7	10	12	15	20
Alarm	4.6±2.5	4.3±2.1	3.9±1.9	3.6±1.6	3.8±2.0	3.6±1.4
Barley	39.5±5.5	38.4±3.9	39.2±4.8	38.6±4.2	39.1±4.4	38.7±3.9
Diabetes	44.8±3.4	44.6±3.7	43.6±3.5	43.7±3.6	43.1±3.6	42.7±3.1
HailFinder	30.2±5.6	30.0±4.4	30.5±4.6	31.1±5.1	30.9±5.6	31.5±5.8
Mildew	18.8±4.2	18.2±4.4	17.3±4.2	16.9±4.0	16.3± 3.8	16.2±3.8
Win95pts	68.8±19.4	68.5±17.8	70.5±21.1	70.0±17.9	70.3±19.9	69.0±20.7

Table 6.7: Mean and standard deviation of the ESHD for ACO-E for each parameter setting (the bold figure is the best for that row)

		ACO-E	GREEDY-E	ACO-B	EPQ
Alarm	SHD	5.9 ± 3.5	21.9 ± 9.0	11.9 ± 11.9	26.1 ± 13.4
	ESHD	4.0 ± 2.0	13.5 ± 6.0	8.1 ± 8.1	18.6 ± 9.7
	BDeu Score ($\times -10^5$)	1.0385 ± 0.0037	1.0389 ± 0.0039	1.0388 ± 0.0038	1.0415 ± 0.0045
Barley	SHD	53.5 ± 6.0	104.8 ± 9.7	67.3 ± 21.6	101.4 ± 14.4
	ESHD	38.9 ± 4.5	78.5 ± 8.3	50.4 ± 16.5	76.9 ± 11.4
	BDeu Score ($\times -10^5$)	5.0708 ± 0.0078	5.2449 ± 0.0124	5.0944 ± 0.0423	5.2354 ± 0.0628
Diabetes	SHD	64.5 ± 5.4	69.2 ± 3.0	70.7 ± 8.5	77.2 ± 7.1
	ESHD	43.7 ± 3.6	46.5 ± 1.9	50.0 ± 6.3	55.1 ± 4.5
	BDeu Score ($\times -10^5$)	1.9392 ± 0.0033	1.9394 ± 0.0033	1.9406 ± 0.0041	1.9457 ± 0.0048
HailFinder	SHD	51.1 ± 7.3	49.1 ± 0.8	74.1 ± 19.7	82.8 ± 18.2
	ESHD	30.7 ± 5.1	28.1 ± 0.7	53.8 ± 14.0	62.9 ± 13.2
	BDeu Score ($\times -10^5$)	4.9206 ± 0.0038	4.9213 ± 0.0036	4.9248 ± 0.0058	4.9481 ± 0.0177
Mildew	SHD	22.7 ± 5.3	29.3 ± 0.7	36.1 ± 14.0	50.3 ± 13.8
	ESHD	17.3 ± 4.2	22.7 ± 0.5	26.7 ± 10.2	39.2 ± 10.7
	BDeu Score ($\times -10^5$)	4.5407 ± 0.0093	4.5531 ± 0.0039	4.5548 ± 0.0170	4.6148 ± 0.0369
Win95pts	SHD	84.5 ± 24.1	104.9 ± 15.5	178.9 ± 58.8	220.1 ± 31.6
	ESHD	69.5 ± 19.5	88.3 ± 12.9	146.7 ± 48.7	184.9 ± 26.6
	BDeu Score ($\times -10^4$)	9.4178 ± 0.0457	9.4649 ± 0.0466	9.4589 ± 0.0717	9.9181 ± 0.0970

Table 6.8: Mean and standard deviation for metaheuristic algorithms (the bold figure is the best for that row)

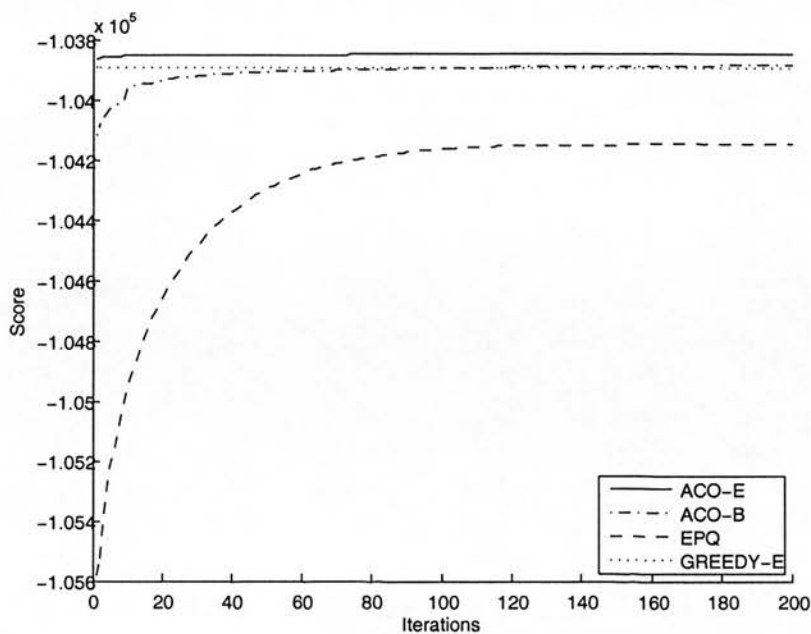


Figure 6.9: BDeu scores for metaheuristic algorithm comparison – Alarm

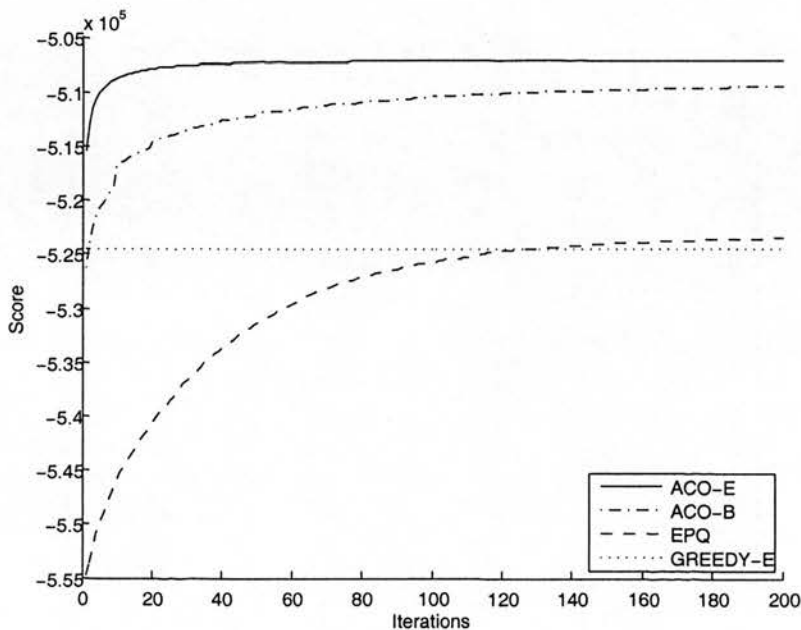


Figure 6.10: BDeu scores for metaheuristic algorithm comparison – Barley

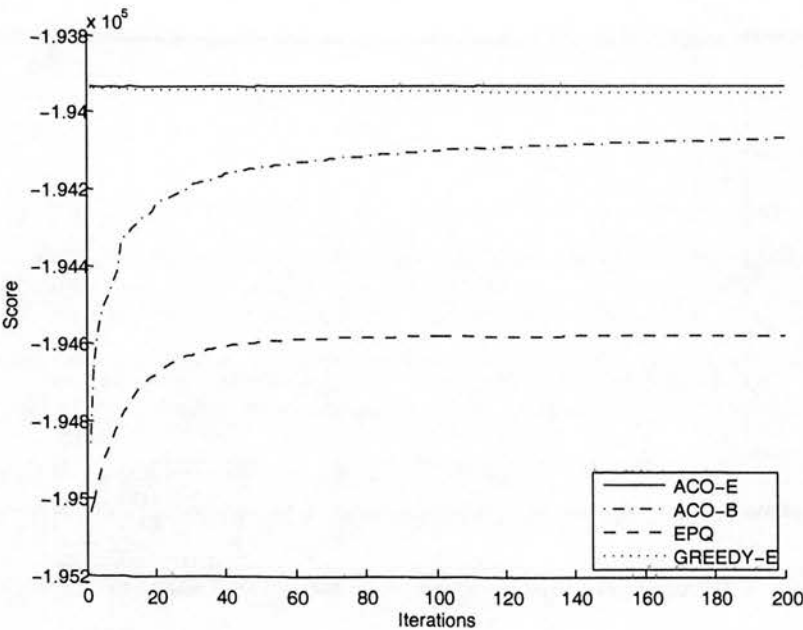


Figure 6.11: BDeu scores for metaheuristic algorithm comparison – Diabetes

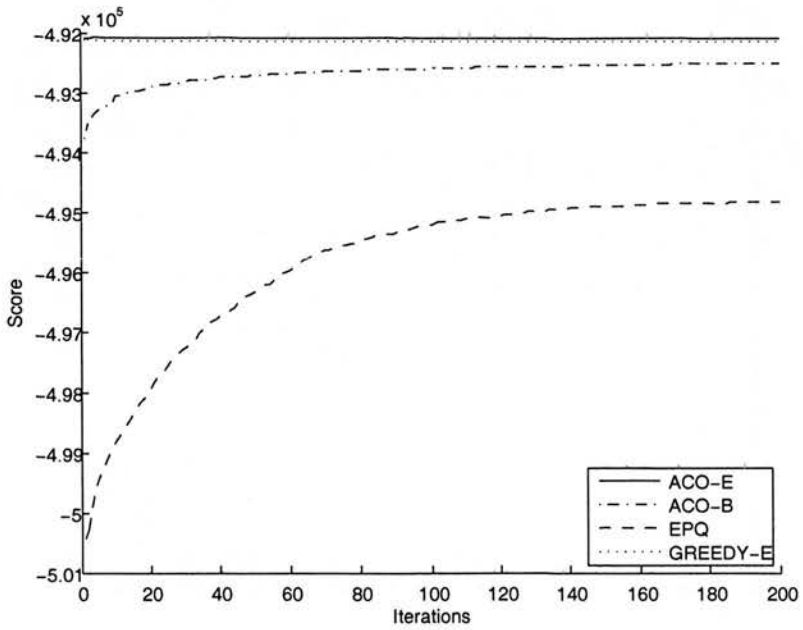


Figure 6.12: BDeu scores for metaheuristic algorithm comparison – HailFinder

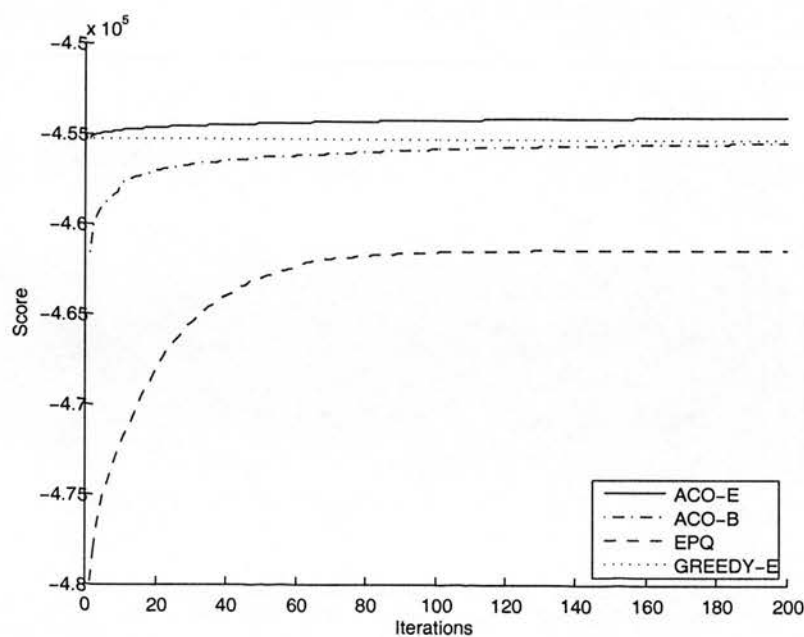


Figure 6.13: BDeu scores for metaheuristic algorithm comparison – Mildew

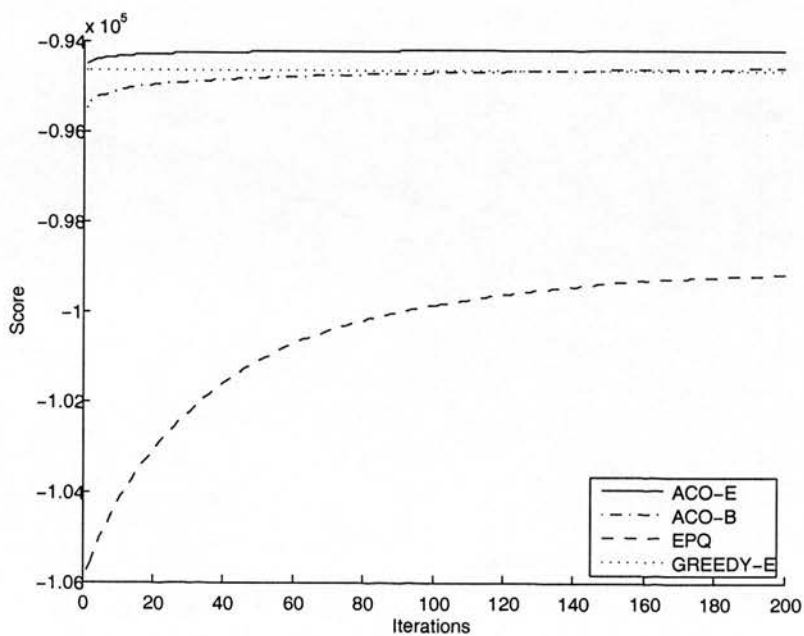


Figure 6.14: BDeu scores for metaheuristic algorithm comparison – Win95pts

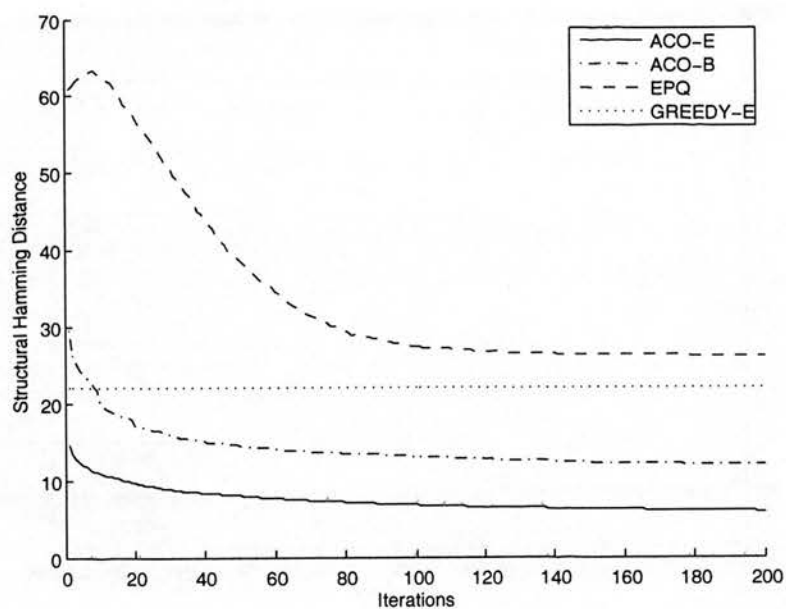


Figure 6.15: SHD for metaheuristic algorithm comparison – Alarm

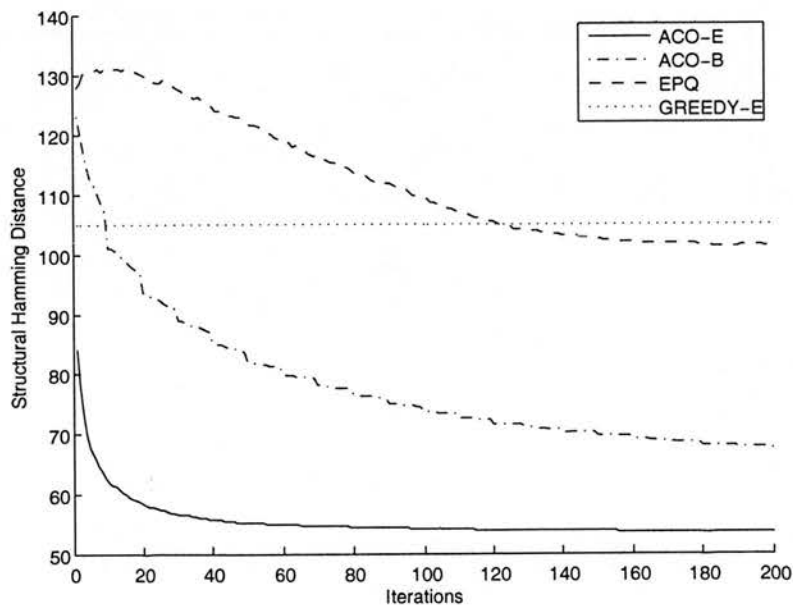


Figure 6.16: SHD for metaheuristic algorithm comparison – Barley

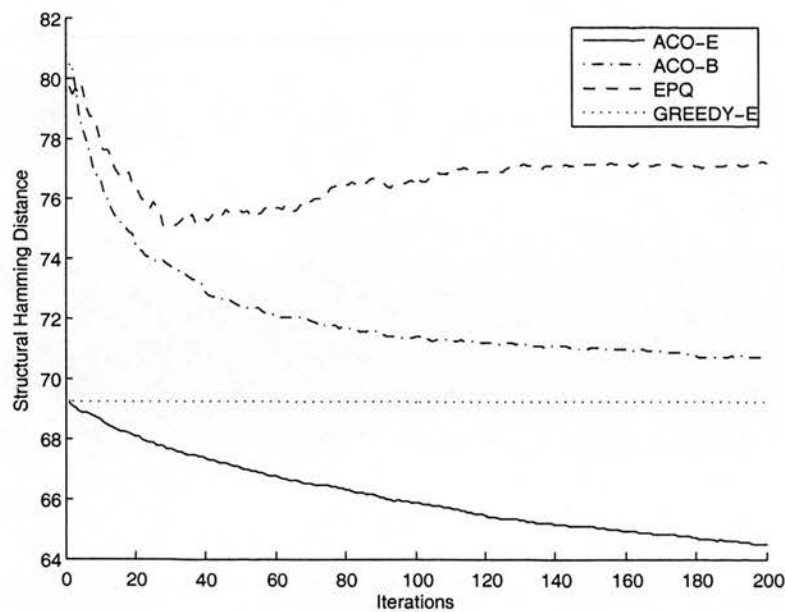


Figure 6.17: SHD for metaheuristic algorithm comparison – Diabetes

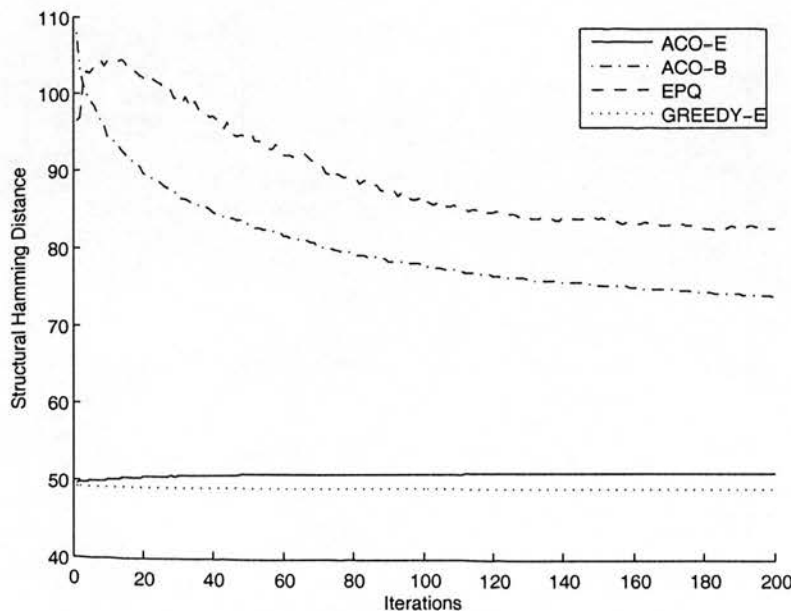


Figure 6.18: SHD for metaheuristic algorithm comparison – HailFinder

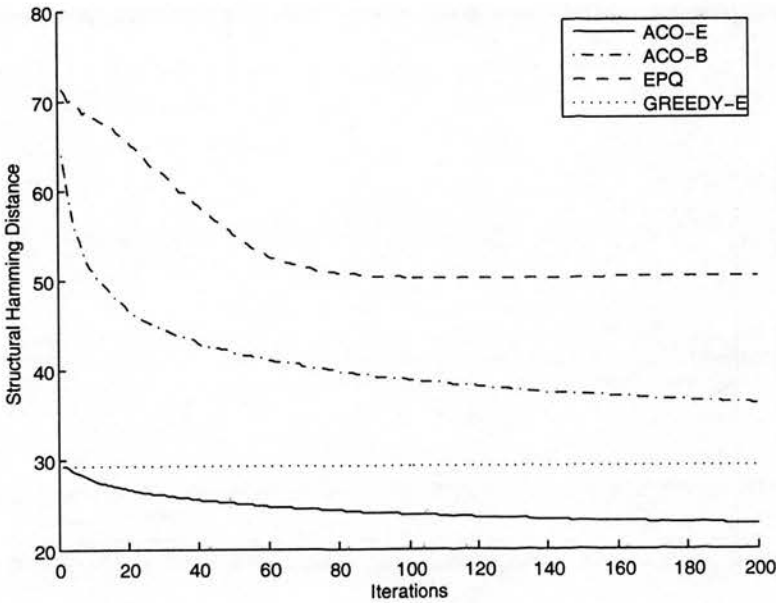


Figure 6.19: SHD for metaheuristic algorithm comparison – Mildew

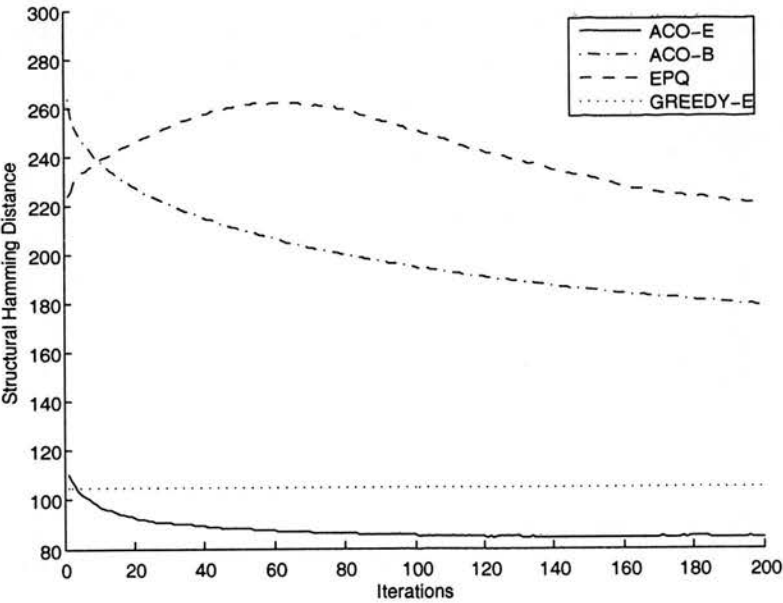


Figure 6.20: SHD for metaheuristic algorithm comparison – Win95pts

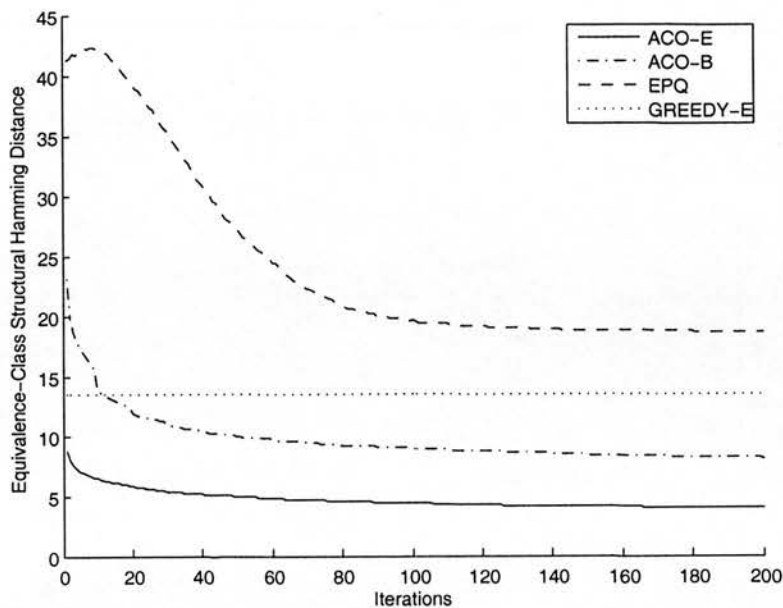


Figure 6.21: ESHD for metaheuristic algorithm comparison – Alarm

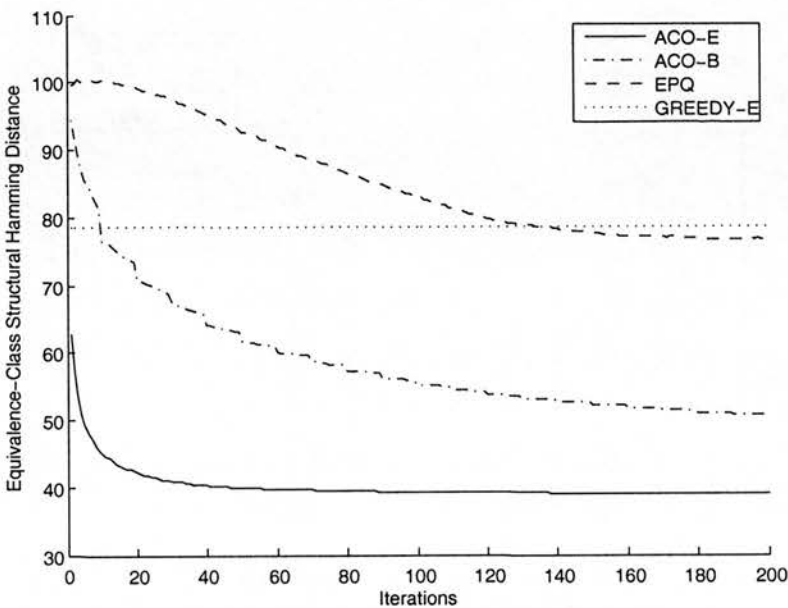


Figure 6.22: ESHD for metaheuristic algorithm comparison – Barley

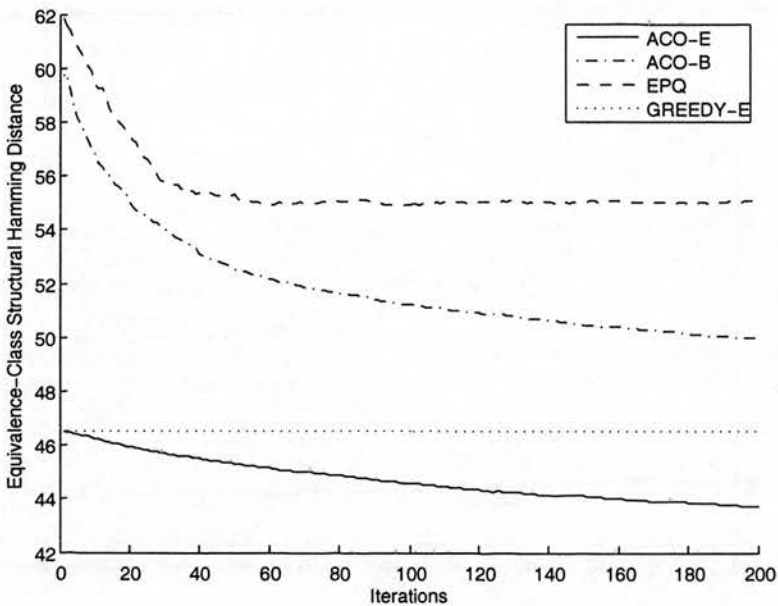


Figure 6.23: ESHD for metaheuristic algorithm comparison – Diabetes

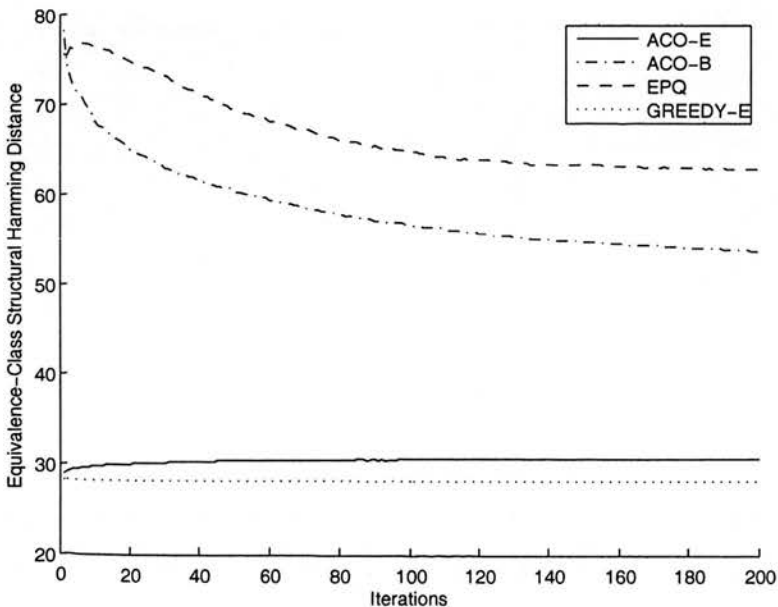


Figure 6.24: ESHD for metaheuristic algorithm comparison – HailFinder

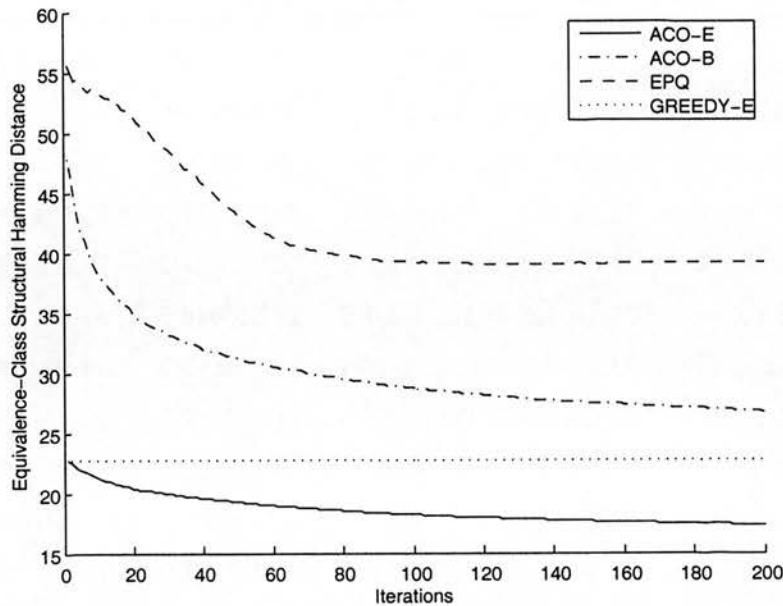


Figure 6.25: ESHD for metaheuristic algorithm comparison – Mildew

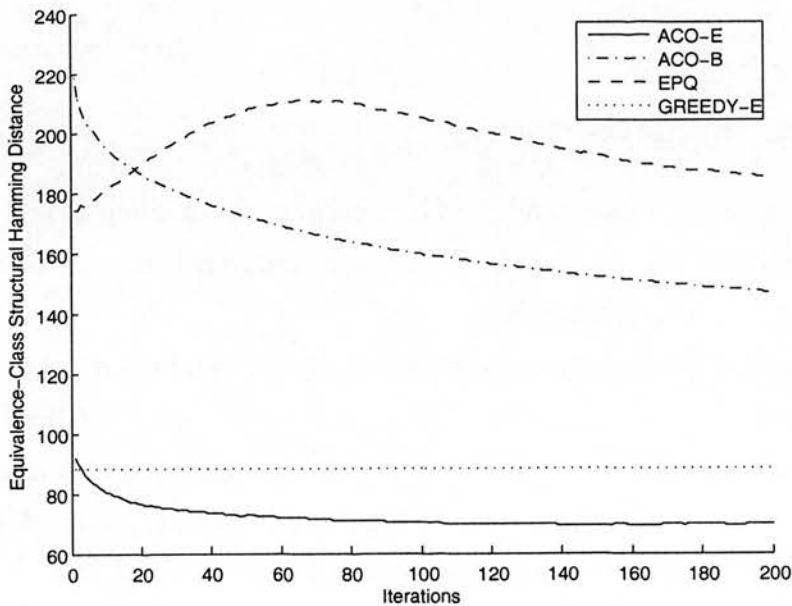


Figure 6.26: ESHD for metaheuristic algorithm comparison – Win95pts

	Alarm	Barley	HailFinder	Mildew
ACO-E	16.4 ± 4.7	80.9 ± 5.3	55.0 ± 5.3	31.0 ± 3.6
MMHC	9.6 ± 7.0	102.6 ± 9.2	208.0 ± 1.6	58.4 ± 7.4
OR1 <i>k</i> = 5	27.8 ± 10.0	109.6 ± 9.5	190.8 ± 14.1	70.6 ± 4.2
OR1 <i>k</i> = 10	31.2 ± 11.1	113.6 ± 15.6	183.2 ± 14.9	75.6 ± 6.3
OR1 <i>k</i> = 20	37.8 ± 9.4	136.4 ± 2.9	184.6 ± 17.2	75.0 ± 4.8
OR2 <i>k</i> = 5	21.2 ± 4.6	120.0 ± 4.5	184.6 ± 14.5	69.2 ± 3.3
OR2 <i>k</i> = 10	33.2 ± 5.4	109.2 ± 16.2	187.0 ± 15.7	64.0 ± 4.4
OR2 <i>k</i> = 20	39.4 ± 6.5	116.8 ± 18.4	200.8 ± 9.2	67.4 ± 3.4
SC <i>k</i> = 5	34.2 ± 3.6	129.6 ± 13.1	194.2 ± 2.5	N/A
SC <i>k</i> = 10	20.4 ± 11.8	N/A	N/A	N/A
GS	58.8 ± 6.5	143.3 ± 7.3	204.2 ± 9.9	62.2 ± 12.2
PC	15.2 ± 1.5	610.0 ± 10.6	385.6 ± 12.5	421.2 ± 10.7
TPDA	9.6 ± 1.5	207.2 ± 4.0	255.4 ± 3.4	97.8 ± 6.8
GES	N/A	159.0 ± 0.0	154.6 ± 54.3	38.8 ± 0.8

Table 6.9: SHD mean and standard deviation for state-of-the-art algorithms (the bold figure is the best for that column)

6.2.1.2 Experimental Condition 2

The results of the experiments conducted to experimental condition 2 are reported here. The second set of comparisons involved ACO-E against other state-of-the-art Bayesian network structure learning algorithms.

The results of this comparison are shown in Table 6.9 and Figures 6.27 to 6.30. The acronyms specified are as given by Tsamardinos et al. (2006) and are indicated in Section 5.3.1.2. Some of the results as supplied by Tsamardinos et al. are missing and are marked by ‘N/A’ in Table 6.9 and are not stated in Figures 6.27 to 6.30. If a result is out of the range of most others, it is represented as a number stating the median. An example of this is in Figure 6.28, where the median of the PC algorithm results is shown as 620. Note that this is different to the value in Table 6.9, where the value 610 is the mean value. The format of the figures is a notched box-and-whisker plot. The centre line in the box is the median, whilst the ends of the box mark the upper and lower quartiles. The whiskers extend to the most extreme value inside 1.5 times the interquartile range,

with the red crosses marking outliers. The notches in the boxes represent the confidence surrounding the median. The length of the notch indicates a 95% confidence interval, calculated as $\text{Median} \pm (1.574 \times \text{Interquartile Range}) / \sqrt{\text{Number of Samples}}$ (McGill et al., 1978). Smaller distances from the top notch to the bottom notch indicate more certainty of the median. If the notches of two boxes do not overlap, then they differ at the 5% significance level.

6.2.2 Discussion

The section will discuss the results presented in the previous section. In general, the discussion will involve looking at the BDeu score, SHD and ESHD values (as defined in Section 5.3.2) obtained by the algorithms. It should be noted that a better BDeu score does not necessarily mean a better SHD or ESHD value and vice-versa. This can occur because of small sample sizes and because of the parameters given to the scoring function (such as the estimated sample size and κ value) which have been shown to produce differences in scoring function behaviour (Kayaalp and Cooper, 2002). In general, different data sets are associated with different parameter values at which the algorithms behave optimally and there does not seem to be a general method to find the optimal values. This problem has been looked at in some depth by Silander et al. (2007).

The first figures to be looked at will be those in Tables 6.4 to 6.7. These presented the results of experiments that varied the parameter values of the ACO-E algorithm. Looking at these figures, there is evidence that the ACO-E algorithm provides behaviour that is better than a simple greedy search for a wide range of values of the parameters.

The next results that will be looked at are Table 6.8 and Figures 6.9 to 6.26. These present ACO-E against other metaheuristic algorithms that are similar. In these results there is strong evidence that ACO-E is performing well against the other algorithms.

Finally, the results given in Table 6.9 and Figures 6.27 to 6.30 will be discussed. These present a series of tests comparing ACO-E to other state-of-the-art Bayesian network structure learning algorithms. Again, looking at the figures, there is strong evidence that ACO-E is competitive in its performance.

In order to perform a comparison, statistical tests will be needed. Because of the non-normality of the distributions of some of the results, tests other than the one shown in Sections 6.1.2.1 and 6.1.2.2 will be used. These are presented below.

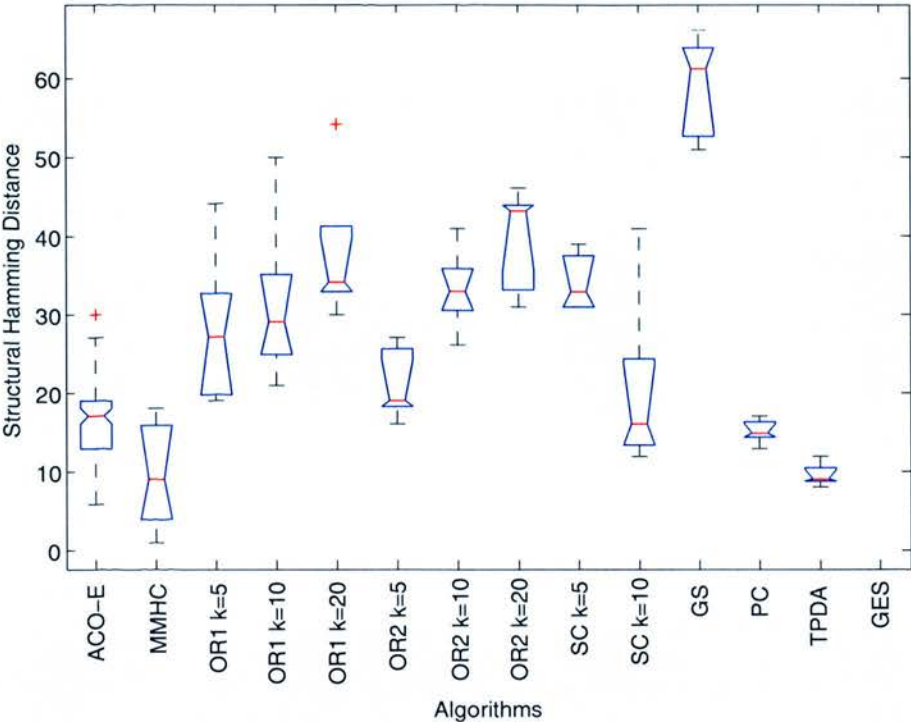


Figure 6.27: SHD results across various state-of-the-art algorithms – Alarm

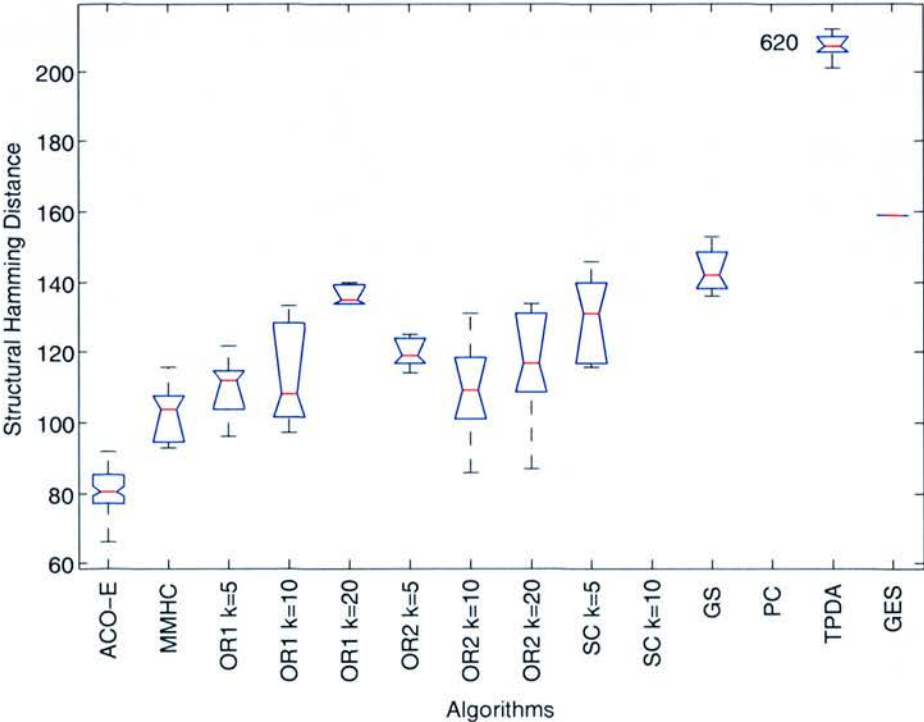


Figure 6.28: SHD results across various state-of-the-art algorithms – Barley

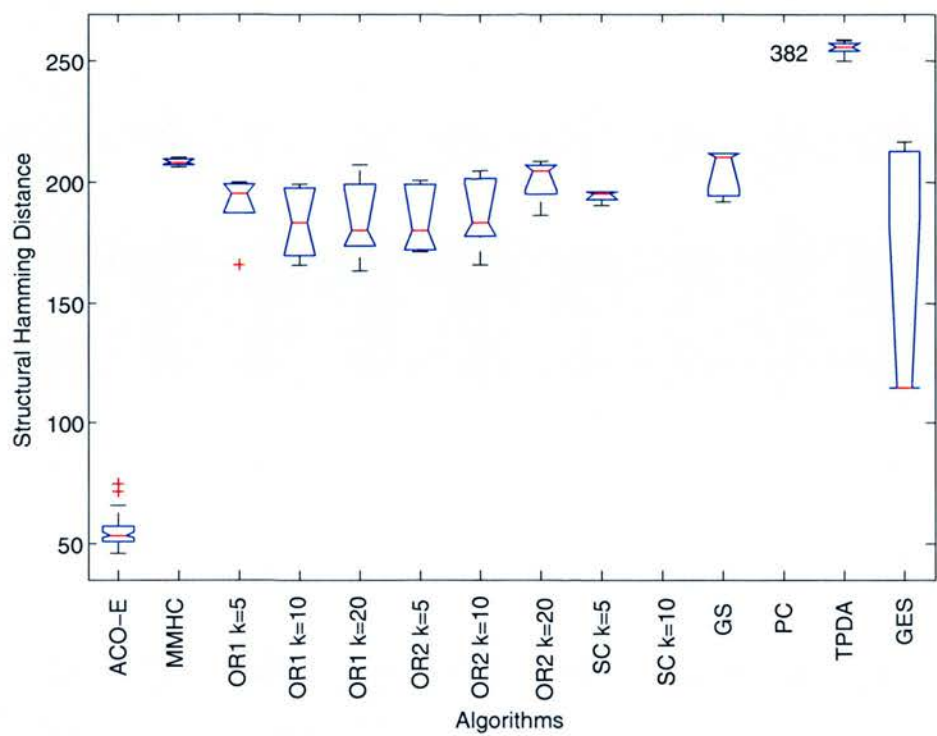


Figure 6.29: SHD results across various state-of-the-art algorithms – HailFinder

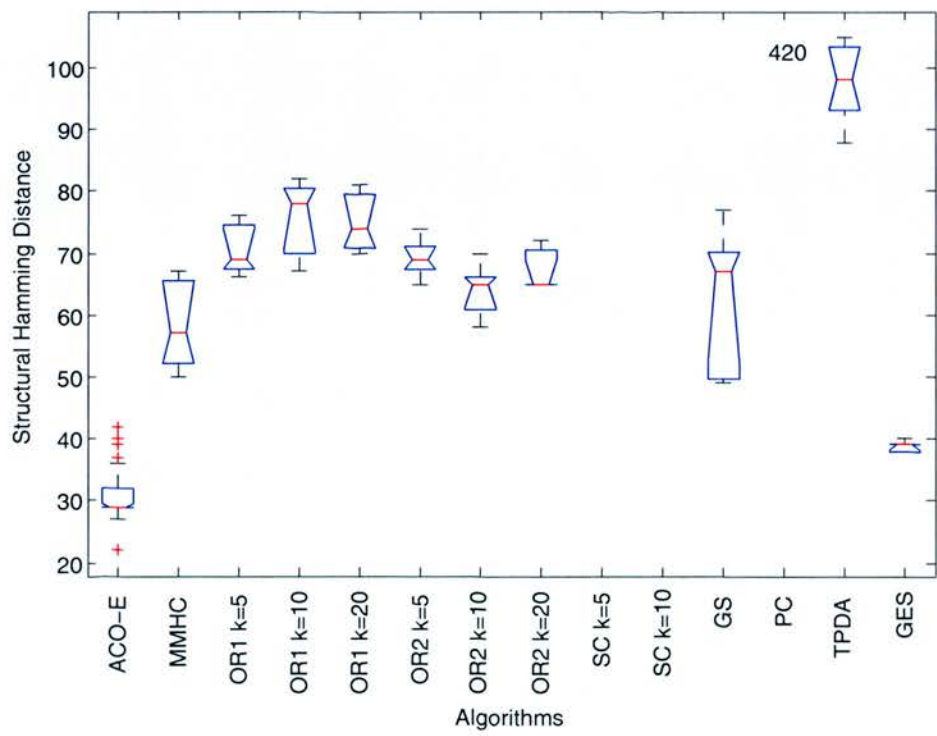


Figure 6.30: SHD results across various state-of-the-art algorithms – Mildew

6.2.2.1 Jarque-Bera Test

The Jarque-Bera Test (Jarque and Bera, 1980) is used to detect whether a given sample of data is normally distributed and is based on the skewness and kurtosis of the data, i.e. how predominant a tail is and how squat the distribution is. A statistic JB is found using

$$JB = \frac{n}{6} \left(S^2 + \frac{(K-3)^2}{4} \right),$$

where S is the sample skewness and K is the sample kurtosis and n is the number of observations. The distribution of JB is asymptotically χ^2 with 2 degrees of freedom. Therefore, given a significance level α , a critical value can be found by looking up the χ^2 table at $\nu = 2$ and α . It can be useful to note that the convergence to the χ^2 distribution is quite slow, so it can be safer to use a special table of JB distribution quantiles.

6.2.2.2 The Mann-Whitney U Test

The Mann-Whitney U test (Mann and Whitney, 1947) (also known as the Wilcoxon (1945) rank-sum test) is a non-parametric test (i.e. it does not assume any particular distribution of the data, such as the normal distribution). In general, it is used to test whether two samples come from the same distribution – in particular, it is used to test whether the means are the same. Firstly consider two samples X and Y . Rank all the data in X and Y together. Let R_X be the sum of the ranks of the X data in the ranking and similarly for Y . A statistic U is calculated for sample X as

$$U_X = R_X - \frac{1}{2} n_X (n_X + 1),$$

where n_X is the number of samples in X and similarly for Y . Given a significance level α and n_X and n_Y , such that $n_X \leq n_Y$, a critical value can be found. The null hypothesis is that $\bar{X} = \bar{Y}$. For a lower one-tailed test, $\bar{X} < \bar{Y}$ if U_X is less than this critical value. For an upper one-tailed test, $\bar{X} > \bar{Y}$ if U_X is greater than the $1 - \alpha$ critical value. For a two-tailed test, the significance level used is $\alpha/2$ and the smaller of U_X and U_Y is checked.

If the sizes of both samples are above 20, then the distribution of U is roughly normal.

6.2.2.3 Conover's Squared Ranks Test

Conover's (1999) squared ranks test is a non-parametric test, that is used to check whether the variances of two distributions are the same. It works in a similar fashion to the Mann-Whitney U discussed above. Instead of ranking the data, the absolute difference from the

mean of the data is ranked, i.e. $|X_i - \bar{X}|$ and $|Y_j - \bar{Y}|$ are ranked together, for all i and j . A statistic T is calculated as the sum of the squares of all the rankings of the variable X , i.e.

$$T = \sum_{i=1}^{n_X} (R(X_i))^2,$$

where $R(X_i)$ is the ranking of datum X_i . Given n_X , n_Y and a significance level α , a critical value can be looked up in a table. To check if $\text{Var}(X)$ is less than $\text{Var}(Y)$, T is checked to be less than this value. If so, then the null hypothesis can be rejected. To check if $\text{Var}(X)$ is greater than $\text{Var}(Y)$, T is checked to be greater than the $1 - \alpha$ critical value. In a two-tailed test, both checks are done, at the $\alpha/2$ and $1 - \alpha/2$ significance levels.

6.2.2.4 ACO-E Behaviour

In this section the behaviour of ACO-E as its parameters are varied will be analysed. As shown in Tables 6.4 to 6.7 there is evidence that there is a difference in the behaviour of the ACO-E algorithm depending on the input parameters. These differences will be analysed using the two-tailed Mann-Whitney U test described above or a standard Student's T test. The particular test used depends on the normality of the data, which can be tested with the Jarque-Bera test.

In order to perform this comparison, the best figures from Tables 6.4 to 6.7 will be compared to the situation where that particular part of the ACO-E algorithm has been turned off. E.g. in Table 6.4 on the Alarm row, the best figure is at $\rho = 0.5$. This is compared to the value at $\rho = 0.0$, as at this value no pheromone deposition or evaporation is occurring. The values at which the various parts of the ACO-E algorithm have been 'turned off' are $\rho = 0.0$, $q_0 = 1$ and $\beta = 0.0$. For the value of $q_0 = 1$, the algorithm behaves purely in a greedy fashion. Therefore for the purposes of testing, the value of the GREEDY-E algorithm in Table 6.8 will be used for comparison, as these results would be exactly the same as the case where $q_0 = 1$. The results of these comparisons are shown in Table 6.10. This table shows p-values for each comparison.

The Behaviour of ρ Looking at Table 6.10 the results for ρ that seem most certain are those for Barley, Mildew and Win95pts. Looking at Tables 6.4, 6.6 and 6.7 for these networks, the best values of ρ are in the 0.2–0.4 range. Also looking at the features of these networks in Table 5.1 there is a correspondence of $\rho = 0.2$ to 76 nodes (Win95pts), $\rho = 0.3$ to 48 nodes (Barley) and $\rho = 0.4$ to 35 nodes (Mildew). Whilst not conclusive, this suggests that ρ behaves well in the region 0.2–0.4 (for those data sets that it works

		ρ	q_0	β
Alarm	SHD	8.0×10^{-4}	4.3×10^{-91}	1.3×10^{-16}
	ESHD	6.5×10^{-5}	3.0×10^{-80}	2.1×10^{-16}
	BDeu Score	1.7×10^{-1}	2.1×10^{-2}	9.1×10^{-3}
Barley	SHD	1.3×10^{-6}	4.2×10^{-230}	6.9×10^{-41}
	ESHD	5.0×10^{-7}	2.0×10^{-182}	1.5×10^{-30}
	BDeu Score	1.3×10^{-9}	2.9×10^{-72}	2.5×10^{-26}
Diabetes	SHD	1.0×10^0	3.6×10^{-42}	8.7×10^{-1}
	ESHD	9.4×10^{-1}	4.8×10^{-39}	8.0×10^{-1}
	BDeu Score	9.2×10^{-3}	2.8×10^{-1}	1.0×10^0
HailFinder	SHD	9.3×10^{-1}	3.3×10^{-2}	5.5×10^{-3}
	ESHD	4.9×10^{-1}	3.3×10^{-10}	3.3×10^{-2}
	BDeu Score	1.9×10^{-1}	1.1×10^{-2}	2.7×10^{-2}
Mildew	SHD	5.1×10^{-16}	3.6×10^{-125}	1.0×10^0
	ESHD	6.8×10^{-15}	8.8×10^{-131}	9.1×10^{-1}
	BDeu Score	1.4×10^{-5}	8.0×10^{-63}	3.5×10^{-1}
Win95pts	SHD	4.9×10^{-8}	9.0×10^{-41}	5.3×10^{-44}
	ESHD	4.2×10^{-8}	1.7×10^{-44}	8.1×10^{-48}
	BDeu Score	1.3×10^{-7}	2.0×10^{-26}	2.2×10^{-18}

Table 6.10: Comparisons of parameter behaviour

at all). As an example consider the Barley values for $\rho = 0.0$ and $\rho = 0.3$ in Table 6.4. At $\rho = 0.0$, there is a mean score of -5.0756×10^5 , with a standard error of $0.0136/\sqrt{216} = 0.0009 \times 10^5$. At $\rho = 0.3$, there is a mean score of -5.0696×10^5 , with a standard error of $0.0039/\sqrt{216} = 0.0003 \times 10^5$. It is obvious from these figures that there is a definite difference between the mean scores.

Looking again at the figures, there is a suggestion that datasets with more nodes would use smaller values of ρ . This makes sense, as larger networks would probably need to spend more time following the best solutions, as a low value of ρ would provide.

The Behaviour of q_0 The parameter q_0 appears to have an effect on most of the networks, with the possible exception of HailFinder. For some of the networks (Alarm and Barley) the parameter has a large effect over a wide range, whereas for others (Diabetes, Mildew and Win95pts), the effect depends to a large extent on the value for q_0 . The largest effects from a scoring function point of view appear to be on the Barley, Mildew and Win95pts networks.

Looking at these networks, the large variations in behaviour across different values of q_0 make it difficult to predict what the best value of the parameter might be for a particular data set. One rule of thumb might be that smaller values of q_0 create more exploration and so might be useful for smaller data sets, whereas larger data sets need more exploitation in order to get to a reasonable answer.

The Behaviour of β From Table 6.10, the networks for which the parameter β plays the most role appear to be Alarm, Barley and Win95pts. Because of the differences of the best values between the scoring function and the SHD it is difficult to predict the best value for β . In the case of Barley, the behaviour is quite robust to values of β in the range 0.5 – 2.5. However, for Alarm and Win95pts, the behaviour depends on the value of the parameter, with a smaller value being better for Alarm and a larger value for Win95pts. As a rule of thumb it appears that networks with fewer numbers of nodes need smaller values of β to help avoid local minima, whereas networks with more nodes need larger values of β in order to focus the search more effectively.

The Behaviour of m Looking at Tables 6.5, 6.6 and 6.7 it can be seen that the value of m can sometimes have a small effect on the effectiveness of ACO-E. In this case, the effect is most pronounced on the Alarm, Diabetes and Mildew networks, with higher values of m giving a smaller SHD and ESHD. Indeed in all cases, higher values of m

never produce statistically worse results, as is to be expected. However, it is important to bear in mind the increased running times with larger values of m .

General Discussion The reason for the strange behaviour of the HailFinder results can possibly be explained by examining its graphs of BDeu score function and SHD/ESHD against time (Figures 6.12, 6.18 and 6.24). It can be seen that as the BDeu score is improving over iterations, the SHD/ESHD value is *deteriorating*. This might lead one to the conclusion that there is a problem with the BDeu scoring function for the HailFinder case, perhaps with its parameters. Another plausible reason for the HailFinder and Win95pts results being out of sync with the others is that they are larger networks, which might favour more aggressive exploitation of the best-so-far solution than the smaller ones. In this case, this would correspond to lower values of ρ and higher values of q_0 . Also heuristic information might be more useful with large numbers of variables, leading to the better results with large values of β . Note that these problems with the HailFinder network have also been seen by de Campos and Castellano (2007).

Comparing the values for SHD and ESHD, it is seen that the optimum values are identical in almost every case; any discrepancies are most likely due to rounding. This lends evidence to the assertion that the ESHD is as at least as useful as the SHD in measuring the quality of reconstructed network structures. Also, in all cases the ESHD value is less than the SHD value. This behaviour is expected from the ESHD's role in seeing how many edits would be needed to turn one structure into another.

6.2.2.5 Behaviour of ACO-E with Respect to Test Network

In the previous section, it was seen that ACO-E can be a useful algorithm in learning the structure of Bayesian networks. It was also seen that the values of the parameters that produced the best behaviour depended on the network that was being tested. Some rules of thumb that consolidate the characteristics observed in the previous section were:

- For data with more variables, use lower values of ρ , higher values of q_0 and higher values of β .
- For data with less variables, use higher values of ρ , lower values of q_0 and lower values of β .

However, it was also seen that ACO-E is not always very successful in learning. This was because little difference was seen when certain parameters were 'turned off' with certain

networks. Looking again at Table 6.10, it seems that the networks for which the effect was most felt were the Barley, Mildew and Win95pts networks. But why is this?

Looking at Table 5.1 there does not seem to be any discernible pattern between the network properties and the suitability of the algorithm. However, the values of the number of v-structures normalised by the number of nodes in the graph, along with the distributions of the nodes across in-degree, as shown in Figures 5.1 to 5.6 show a more definite reason. The networks with which ACO-E performed well all have a larger number of nodes that have a higher in-degree. As a result of this, data sampled from these networks is better going to match a similar network in the scoring function, i.e. one that is similar to the standard network. Because the search starts from the empty graph, it is more likely that the search would get trapped in a local minimum in trying to add enough arcs to get to the needed number. Due to ACO-E being a stochastic algorithm, it is able to avoid these local minima.

The upshot of this is that ACO-E would be a good candidate algorithm for data that contains variables with a large degree of dependence on many other variables.

6.2.2.6 Metaheuristic Algorithm Comparison

Figures 6.9 to 6.26 and Table 6.8 show the results of comparing ACO-E against other metaheuristic algorithms. It can be seen that ACO-E performs better than the other algorithms shown, except in the case of the HailFinder network, where GREEDY-E gives a better result for the SHD/ESHD. However, in this case, ACO-E gives a better BDeu score value. This is the same as the problem discussed in Section 6.2.2.4, that gave a better score for a worse structure.

These statements can be backed up by looking at Table 6.11 which gives p-values for a two-tailed Mann-Whitney U test comparing ACO-E results against the other algorithms after runs had ended. With this statistic, the smaller the number, the more significant the test. Since the results from each of the runs come from a separate sample of the network, the correct tests would be unpaired. The data used in the tests were those from the metaheuristic algorithm comparison, i.e. over all combinations of the parameters for ACO-E and ACO-B. It seems that in most cases, the results are highly significant, which supports the assumption that ACO-E performs well. In the cases where the significance is not so high (ACO-E BDeu score compared to GREEDY-E BDeu score with the Alarm network and ACO-E score compared to GREEDY-E BDeu score with the Diabetes network), it should be noted that tiny changes to the BDeu score value can lead to large structural changes as an algorithm converges towards the optimum (generating) network.

In these cases, the SHD and ESHD p-values show a highly significant difference.

Comparisons were also made between the variances of the results as seen in Table 6.12, which gives p-values for Conover's Squared Ranks one tailed test. From this table it can be seen that ACO-E generally has a lower standard deviation in its results after finishing its run compared to ACO-B and EPQ. Whilst the standard deviation of results compared to GREEDY-E are significantly lower with respect to the Alarm and Barley networks, in the other cases GREEDY-E seems to be the most consistent with regards to its final results.

It should be noted that non-parametric tests were used, as the results in general, had non-normal distributions. It should also be noted that some of the results in the tables might seem incorrect. E.g. in Table 6.11, in the Win95pts-Score row, the test for ACO-B is more significant than that for GREEDY-E, even though the mean of GREEDY-E is further from ACO-E than that of ACO-B in Table 6.8. This is because of the non-normality of the data, the main reason that non-parametric tests were used.

6.2.2.7 State-of-the-art Algorithm Comparison

In this section, the comparison of ACO-E against other state-of-the-art Bayesian network structure learning algorithms will be analysed. As shown in Table 6.9 and Figures 6.27 to 6.30, ACO-E appears to have good performance against these other algorithms. The results of statistical comparisons of ACO-E against these algorithms are shown in Table 6.13.

In this table are shown p-values for individual comparisons of ACO-E against the other algorithms. The test used for all these comparisons was the Mann-Whitney U test. This test was used, as the distributions were found to be not normal. At the foot of the table is the combined p-value found from the individual p-values above it. This is the total p-value for comparing ACO-E against all the other algorithms. The method of combining these values was

$$p_{combined} = 1 - \prod_{i=1}^n (1 - p_i),$$

where p_i is the p-value of entry i in the table, there being n values in total. This method of combining the p-values is needed because of the chance of causing a Type I error otherwise. A Type I error is a false positive result, i.e. the null hypothesis is rejected when it should not be. This can occur in this case because if an experiment with a small chance of failing is repeated enough times, there will be a large chance that at least one of them

		GREEDY-E	ACO-B	EPQ
Alarm	SHD	1.1×10^{-113}	1.6×10^{-31}	1.9×10^{-118}
	ESHD	2.0×10^{-112}	2.5×10^{-34}	1.5×10^{-126}
	Score	4.3×10^{-2}	6.0×10^{-3}	1.8×10^{-18}
Barley	SHD	9.6×10^{-124}	1.8×10^{-92}	9.3×10^{-123}
	ESHD	6.3×10^{-124}	5.4×10^{-126}	2.6×10^{-123}
	Score	5.0×10^{-123}	5.9×10^{-96}	7.0×10^{-123}
Diabetes	SHD	3.1×10^{-34}	4.7×10^{-104}	1.5×10^{-88}
	ESHD	1.4×10^{-30}	6.7×10^{-188}	6.6×10^{-114}
	Score	2.5×10^{-1}	2.3×10^{-18}	4.5×10^{-69}
HailFinder	SHD	3.8×10^{-12}	2.7×10^{-237}	1.8×10^{-99}
	ESHD	5.9×10^{-20}	0	5.5×10^{-119}
	Score	4.4×10^{-3}	2.2×10^{-93}	4.6×10^{-113}
Mildew	SHD	2.3×10^{-50}	4.7×10^{-160}	7.5×10^{-115}
	ESHD	7.2×10^{-57}	1.4×10^{-134}	1.9×10^{-116}
	Score	7.4×10^{-62}	5.7×10^{-114}	5.3×10^{-118}
Win95pts	SHD	1.5×10^{-42}	0	4.8×10^{-123}
	ESHD	5.6×10^{-49}	0	4.7×10^{-123}
	Score	3.0×10^{-37}	4.1×10^{-53}	5.0×10^{-123}

Table 6.11: p-values for Mann-Whitney *U* test, 10,000 samples

		GREEDY-E	ACO-B	EPQ
Alarm	SHD	3.6×10^{-84}	2.4×10^{-252}	5.3×10^{-104}
	ESHD	4.1×10^{-103}	0	7.3×10^{-117}
	Score	8.0×10^{-2}	1.3×10^{-1}	5.0×10^{-5}
Barley	SHD	1.3×10^{-61}	1.5×10^{-274}	2.3×10^{-66}
	ESHD	6.2×10^{-44}	7.6×10^{-277}	1.6×10^{-69}
	Score	1.5×10^{-66}	0	4.6×10^{-146}
Diabetes	SHD	1	1.5×10^{-3}	1.0×10^{-4}
	ESHD	1	1.2×10^{-13}	3.1×10^{-5}
	Score	4.8×10^{-1}	2.1×10^{-4}	7.3×10^{-8}
HailFinder	SHD	1	2.0×10^{-153}	2.6×10^{-62}
	ESHD	1	2.5×10^{-169}	2.0×10^{-58}
	Score	9.1×10^{-1}	4.3×10^{-23}	1.3×10^{-143}
Mildew	SHD	1	1.5×10^{-111}	9.3×10^{-54}
	ESHD	1	1.3×10^{-77}	1.9×10^{-49}
	Score	1	1.5×10^{-85}	8.3×10^{-79}
Win95pts	SHD	1	5.7×10^{-209}	5.4×10^{-12}
	ESHD	1	8.7×10^{-210}	1.4×10^{-13}
	Score	5.8×10^{-1}	2.6×10^{-26}	3.2×10^{-38}

Table 6.12: p-values for Conover's squared ranks test, 10,000 samples

	Alarm	Barley	HailFinder	Mildew
MMHC	3.2×10^{-2}	2.1×10^{-8}	2.1×10^{-8}	2.1×10^{-8}
OR1 $k = 5$	5.9×10^{-4}	2.1×10^{-8}	2.1×10^{-8}	2.1×10^{-8}
OR1 $k = 10$	1.9×10^{-5}	2.1×10^{-8}	2.1×10^{-8}	2.1×10^{-8}
OR1 $k = 20$	4.1×10^{-8}	2.1×10^{-8}	2.1×10^{-8}	2.1×10^{-8}
OR2 $k = 5$	3.5×10^{-2}	2.1×10^{-8}	2.1×10^{-8}	2.1×10^{-8}
OR2 $k = 10$	1.4×10^{-7}	1.1×10^{-5}	2.1×10^{-8}	2.1×10^{-8}
OR2 $k = 20$	2.1×10^{-8}	3.8×10^{-6}	2.1×10^{-8}	2.1×10^{-8}
SC $k = 5$	2.1×10^{-8}	2.1×10^{-8}	2.1×10^{-8}	N/A
SC $k = 10$	8.1×10^{-1}	N/A	N/A	N/A
GS	2.1×10^{-8}	4.3×10^{-7}	2.1×10^{-8}	2.1×10^{-8}
PC	4.0×10^{-1}	2.1×10^{-8}	2.1×10^{-8}	2.1×10^{-8}
TPDA	6.5×10^{-4}	2.1×10^{-8}	2.1×10^{-8}	2.1×10^{-8}
GES	N/A	2.1×10^{-8}	2.1×10^{-8}	2.1×10^{-6}
$p_{combined}$	3.3×10^{-2}	1.6×10^{-5}	2.5×10^{-7}	2.3×10^{-6}

Table 6.13: p-values comparing ACO-E against state-of-the-art algorithms

will fail. It should be noted that the value at the foot of Alarm does not combine all the p-values above it. Instead it leaves out those of ‘SC $k = 10$ ’, ‘PC’ and ‘OR2 $k = 5$ ’. Using these results would have resulted in a high combined p-value. Therefore, it could not be said that the ACO-E results are significantly different from the three results mentioned. Note that combining the p-values in this manner is not intended to produce a very accurate figure whether the results are different or not. It is more intended to provide an illustration of the chance a Type I error occurred. In this case, leaving out the three aforementioned results is valid as they are explicitly not being compared against.

The results given in Table 6.9 and Figures 6.27 to 6.30 appear to be indicative of the results as given in Section 6.2.2.4. As discussed there, ACO-E had some effect with learning in the Alarm network, especially against a straight greedy search. However, most of this effectiveness appeared to come from the randomness of the search, and did not make much use of the ρ and β parameters.

On the networks which ACO-E performed well on, Barley and Mildew, this performance is reflected across to the current results as it also performed well here. The results for the HailFinder network seem surprising, as the ACO-E algorithm did no better than

a search using GREEDY-E. However, in this figure (6.29), the performance of GES can also be seen to be doing well. As GES works in the space of equivalence classes of Bayesian networks, it is postulated that ACO-E performs well because of the structure of the search space.

As ever, comparisons must be taken tentatively, especially in this case, as the results given by Tsamardinos et al. only have five samples.

6.3 Summary

In this chapter, two main topics were presented; results from the experiments described in Chapter 5 and a discussion of these results. Similar to previous chapters, two threads ran through these topics. These were speeding up the learning of Bayesian network structures as described in Section 4.2 and the ACO-E algorithm as described in Section 4.3.

Firstly the results and discussion of the speeding-up methods were illustrated. The results suggested a possible improvement in the running time of the ACO-E algorithm. These suggestions were tested statistically and the results indicated a strong rejection of the hypothesis that there was no improvement. Other tests were run that attempted to quantify the speed-up found. It was seen that the speed-up depends on the number of variables being dealt with. Bringing the theoretical results of Section 4.2.1 into play, it was shown that the running time was a constant function of the square root of the running time of the algorithm without the speeding-up methods.

After this, the results and discussion of the ACO-E thread were given. The results of the experiments with conditions 1 and 2 as presented in Sections 5.3.1.1 and 5.3.1.2 were presented in three parts. These were to enable comparison of the behaviour of the ACO-E algorithm as its parameters were varied, comparison of the ACO-E algorithm against other metaheuristic algorithms that were similar and comparison of the ACO-E algorithm against other state-of-the-art Bayesian network structure learning algorithms.

From the first comparison, certain traits of ACO-E were found. It was seen that ACO-E does not always work well on all data sets; it performed well on those data sets that had a high degree of connectivity in a large number of variables. It was postulated that this is due to local maxima in the learning surface. Being a stochastic algorithm, ACO-E was able to work around these. The behaviour of the various parameters was also looked at and it was found that in data sets with many features, ACO-E performed well with low values of ρ , high values of q_0 and high values of β .

ACO-E compared well to other metaheuristic algorithms, doing better in all cases. It also performed well against other state-of-the-art algorithms, doing better against many of them by a large margin. Part of this performance was attributed to the properties investigated in Section 6.2.2.5 and also to the search space that ACO-E is defined in.

Chapter 7

Modelling the Circadian Clock of *Arabidopsis Thaliana* using a Dynamic Bayesian Network

THE ANALYSIS OF DATA obtained from experiments looking at the expression of genes has become an important topic in the realm of bioinformatics. Whilst other bioinformatics tasks such as sequence alignment, gene finding, genome assembly and genome annotation have proven very amenable to computational analysis, there are tasks which are intrinsically difficult at the current level of knowledge. These include protein structure prediction and modelling gene regulatory networks. The latter will be examined in this chapter.

These gene regulatory networks will be modelled by using a modified version of ACO-E to build dynamic Bayesian networks (DBNs). DBNs, as explained in Section 2.1.4.2, are an extension of Bayesian networks. They consist of a prior network, used to model initial conditions, and a transition network, used to model the effect of interactions over time. The set of operators used in the ACO-E algorithm will be slightly modified in order to generate a valid DBN. Whilst DBNs have been used to model gene regulatory networks by Husmeier (2003), Kim et al. (2003), Perrin et al. (2003), Zou and Conzen (2005) and others, the method presented here differs in how the data is treated and in how it can infer interactions at multiple time periods.

With these modifications in place, a set of gene expression data obtained from experiments observing particular genes of the plant *Arabidopsis thaliana* will be used. These genes are known to behave in a clock-like fashion that regulates the function of the plant.

Experiments will be conducted to learn the structure of DBNs using ACO-E. These networks will be compared to a standard network developed by an expert in the field being analysed. Using these results, the behaviour of ACO-E in constructing gene-regulatory networks will be investigated.

7.1 Background

Bioinformatics is the marriage of the two fields of biology and informatics. Algorithms and statistical techniques from various subfields of informatics such as machine learning are applied to problems in the biology domain that would take much longer if done by hand. These problems include:

Sequence alignment Arranging sequences of DNA from the the genes of organisms in order to overlap similar patterns, i.e. to match the sequences as much as possible. This can enable the identification of the functionality of a gene given that the functionality of the other gene is known.

Gene prediction Finding out the areas of a gene that are functional and in some cases, predicting what functionality they have.

Genome assembly Assembling a number of DNA sequences into a single coherent sequence.

Protein structure prediction Anticipating the structure of a protein (and hence its functionality) from the sequence of amino acids that compose it.

Genome expression analysis Predicting how genes are expressed (that is, how they turn 'on' and 'off' with respect to how other genes are 'on' or 'off'). With many genes, this turns into the study of networks of genes that interact with each other. These are referred to as gene regulatory networks.

These problems and others are explained in many introductory texts on the subject (Lesk, 2002). The analysis of gene regulatory networks is a hard task. In order to see how various genes interact, experiments must be performed which measure the expression levels of these genes. Using this data, it is possible to develop models of genetic systems behaviour. One of the earliest and still commonly used techniques is to use differential equations that show how expression levels vary with respect to other expression levels.



Figure 7.1: *Arabidopsis thaliana*

However, recent advances in measurement of gene expression levels, using techniques such as DNA microarrays, have allowed the measurements of tens of thousands of genes to be performed simultaneously. With this amount of genes in an organism, all possible interactions between all the genes cannot be looked at. Therefore, methods involving graphical representations have become more widely used. These include Boolean networks, networks similar to neural networks, stochastic process calculi and Bayesian networks.

7.1.1 Data from *Arabidopsis thaliana* Experiments

The experimental study in this chapter will involve data obtained from experiments on *Arabidopsis thaliana*, a small flowering plant shown in Figure 7.1. It has long been used as a model plant for research (Meyerowitz, 2001), and was the first plant to have its genome sequenced, by The Arabidopsis Genome Initiative (2000).

The particular experiments in question were designed to study the ‘plant circadian rhythms’ of *Arabidopsis thaliana*, i.e. the oscillating behaviour of plants as they respond to the change in sunlight (McClung, 2006). Briefly, the expression levels of the genes in plants tend to synchronise with the rising and setting of the sun, oscillating between

CONDITION	AT0029	AT0030	AT0031b	AT0032	AT0033	AT0047
RATIO	6:18	9:15	12:12	15:9	18:6	3:21

Table 7.1: Ratio of light to darkness for each experimental condition

high and low levels. However, when the light source is removed, the expression levels continue to oscillate with the same frequency, with the behaviour decaying over time.

In the experiments, there were in general two phases. The first phase, called the entrainment, switched the light on and off at regular intervals and lasted three days. The second phase had constant light and also lasted three days. At intervals of 1.5 hours, readings were taken of the expression levels of the genes being investigated.

The experiments involved ten different genes: CAB, GI, CCA1, CCR2, LHY, CAT3, ELF3, PRR9, TOC1 and ELF4. Further information about these genes can be found in (McClung, 2006). There were also six different conditions under which the experiments were conducted; each condition had a different light period for the entrainment phase of the experiment. Table 7.1 shows, for each experimental condition, the ratio of light to dark for that condition. E.g. for condition AT0029, there are 6 hours of light followed by 18 hours of darkness in the entrainment phase. Because the experiments lasted six days, and a sample was taken every 1.5 hours, there were a total of 96 samples taken for each condition. The first of these was taken as a reference datum and subtracted from each of the following samples, leading to 95 samples with which to work.

7.2 Modelling Gene Expression Data using DBNs

It is the stochastic nature of the gene expression process that makes probabilistic models an appropriate choice in modelling them. One interesting approach is in using Bayesian networks as a probabilistic model. This enables large scale, non-linear interactions between many genes to be represented and simulated. E.g., recent studies by Huang et al. (2007) have looked at learning Bayesian networks with thousands of genes. The large amounts of data present in microarray studies can be used to learn both the structure and parameters of the network – missing values and noise can also be taken care of. Also, Bayesian networks can be given a causal interpretation. A problem arises however in that loops are not permitted in Bayesian networks. Therefore, feedback processes cannot be represented.

A solution to this problem is to use dynamic Bayesian networks. With DBNs, it is

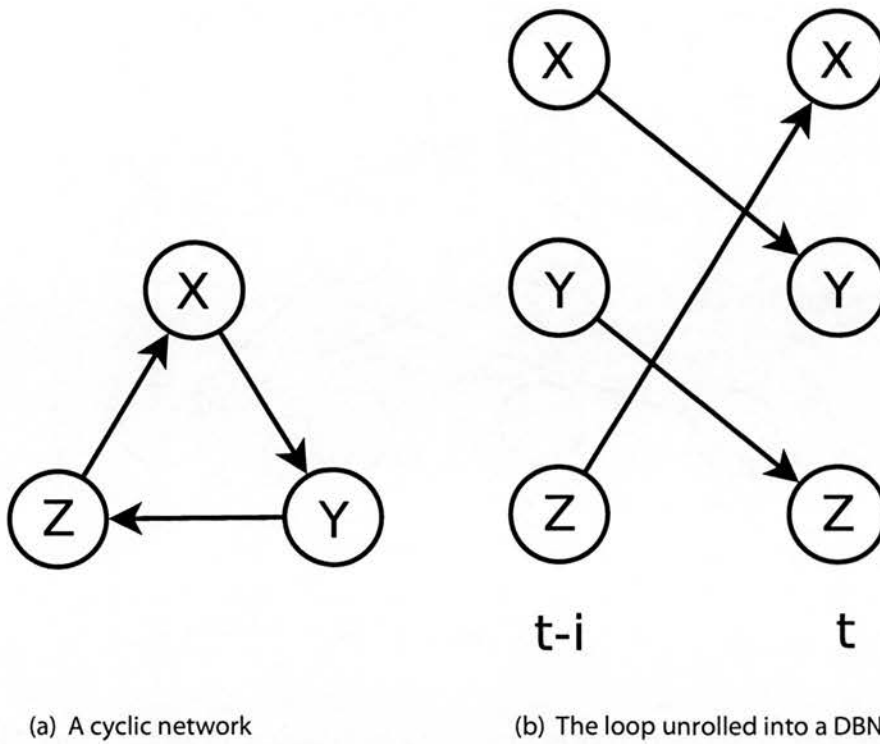


Figure 7.2: A Bayesian network unrolled to a dynamic Bayesian network

possible to ‘unroll’ what would be a loop in a Bayesian network across time. E.g. Figure 7.2 shows a network with a loop being unrolled. Here, the loop involving the variables X , Y and Z has been transformed into a dynamic Bayesian network, with a time delay of i . Hence, a feedback loop has been established, with a period of $3i$. Note that the looped network in Figure 7.2 is not properly a Bayesian network, but a representation of what is wanted (i.e. a feedback loop amongst variables).

7.2.1 Using ACO-E to Learn a Dynamic Bayesian Network

Like most Bayesian network structure learning algorithms, ACO-E is designed to learn a static structure. Whilst dynamic Bayesian networks are very similar to static networks, there are slight differences that mean extra care has to be taken when trying to learn them. Bearing in mind that a DBN has two parts, the prior and transition network, this section will focus on learning the transition network; the prior network is exactly the same as a normal Bayesian network.

In a transition network, the nodes are grouped into multiple layers that designate different timepoints. Arcs can never go back in time and the head of any arc can only be in a single layer, normally that layer at time t . It is very normal to have two layers, such

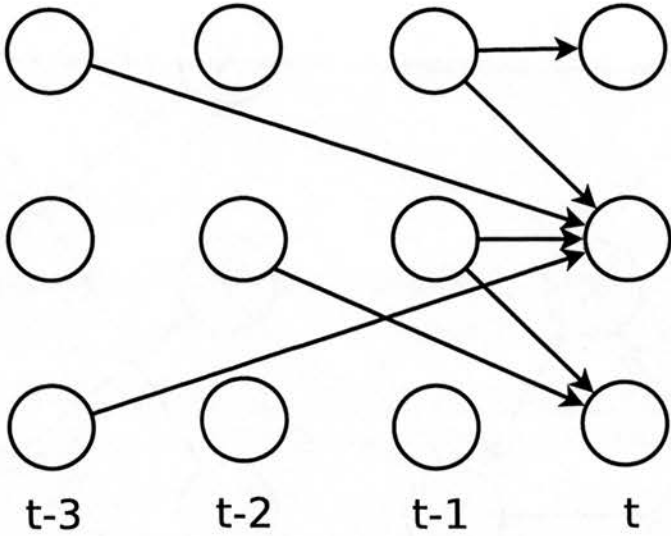


Figure 7.3: A 4-layer dynamic Bayesian network

that they correspond to variables at time t and $t - 1$. In fact, in all the general literature on Bayesian networks surveyed, learning DBNs amounted to learning DBNs with two layers. However, in general, $n + 1$ layers can be specified, from $t - n$ to t . An example of a 4-layer DBN is shown in Figure 7.3.

When learning such a DBN, it is sufficient to add an extra constraint to the learning algorithm, i.e. the head of an arc must always be in the layer at time t . By following this, any Bayesian network structure learning algorithm that searches through the space of DAGs can be used and will produce a valid network. Things are more difficult when searching through E-space. E.g. it does not make sense to insert an undirected arc between the nodes in two layers, as this implies the arc could be directed either way, whereas it must be directed forward in time. Therefore, extra conditions must be attached to each of the operators that are being used to traverse E-space. For each of the operators defined in Table 4.2 the extra conditions are shown in Table 7.2.

Another change that must be made whilst searching for a DBN through E-space is to change the DAG-to-CPDAG procedure. This method, as explained by Chickering (2002a), is used as a subroutine in a search, after an operator has been applied and the PDAG-to-DAG subroutine has been called. It is used to turn a DAG into the CPDAG representing the equivalence class of that DAG. When used with a dynamic Bayesian network, it must be modified so that edges that go forward in time in a DAG are labelled as compelled in order that they go forward in time in the CPDAG.

Finally, because each datum in the data set corresponds to a different time point, the

Operator	Conditions
InsertU $x - y$	x and y must be elements of the layer at time t
DeleteU $x - y$	No extra conditions
InsertD $x \rightarrow y$	<ul style="list-style-type: none">y must be an element of the layer at time tIf x is not an element of the layer at time t, no other conditions need be checked
DeleteD $x \rightarrow y$	No extra conditions
ReverseD $x \rightarrow y$	x must be an element of the layer at time t
MakeV $x \rightarrow z \leftarrow y$	No extra conditions

Table 7.2: Extra conditions for operators in a dynamic Bayesian network

scoring function must also look back in time in order to capture past behaviour. E.g. with the DBN in Figure 7.3 the scoring function must look at the data at the current row and each row up to three rows back. This can easily be achieved by duplicating data with variables X_1 to X_n to extra variables and offsetting the data forward corresponding to the offset from t in the DBN. E.g. with the DBN in Figure 7.3 there would be extra variables X_{n+1} to X_{2n} , X_{2n+1} to X_{3n} and X_{3n+1} to X_{4n} . The first set of extra variables would have the data offset by 1 datum, the second by 2 data and the third by 3 data.

With these changes in place, ACO-E can be used to conduct a search for a DBN in E-space, without any differences in the algorithm itself.

7.3 Experimental Methodology

In order to test the ability of ACO-E to learn a genetic regulatory network, experiments were conducted using the *Arabidopsis thaliana* data discussed in Section 7.1.1. This section will describe the steps taken to achieve this. This includes preprocessing the data, formulating prior knowledge, designing a methodology to run the algorithm and testing the results obtained against expert knowledge.

7.3.1 Preprocessing the Data

The data obtained from the experiments described in Section 7.1.1 are continuous in nature, i.e. they show the level of gene expression as measured. However, most scoring functions for use with structure learning are based on nominal data, i.e. data that is of a discrete nature. For the purposes of these experiments, the BDeu scoring function as described in Section 4.1 was used. For this to be utilised, the data has to be discretised.

Whilst discretising the data is an easy option, there is a problem with this as shown by Figure 7.4. It can be seen that as the expression level decays after entrainment, the discretised expression level flatlines, even though there is still oscillatory action occurring. One way to counter this is to discretise the data into multiple levels. However, in doing so, the amount of data at each level is reduced. This can lead to less support for dependencies when a model is fitted to the data, something that should be avoided when there is not much data to begin with.

In order to avoid this problem the first derivative of the data was taken and the data discretised as to whether this derivative was greater than or less than zero. The results of this procedure are shown in Figure 7.5. It can be seen that this discretisation captures the

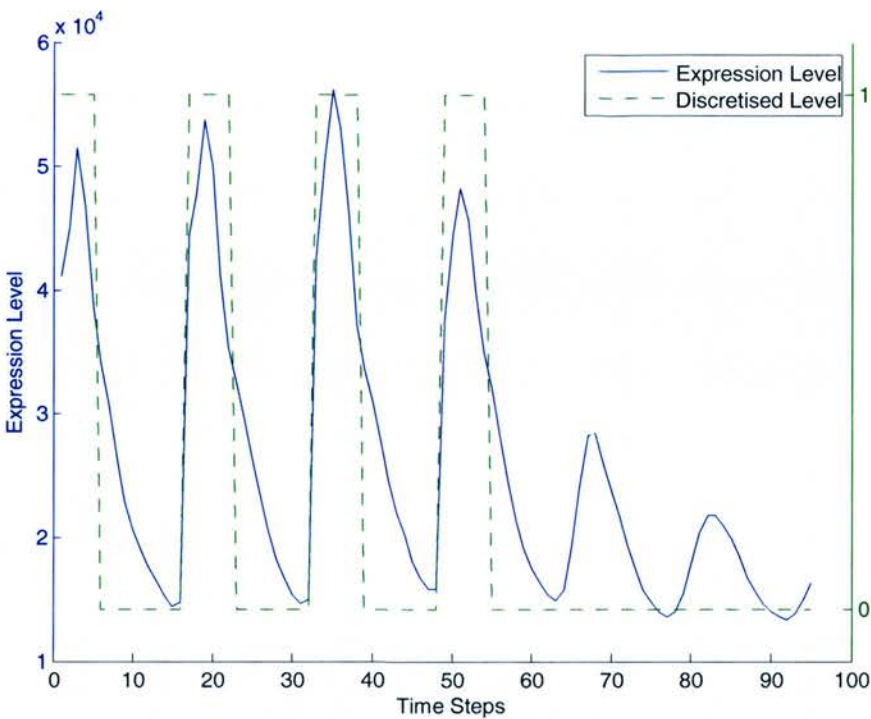


Figure 7.4: Expression level of PRR9 in the AT0029 condition

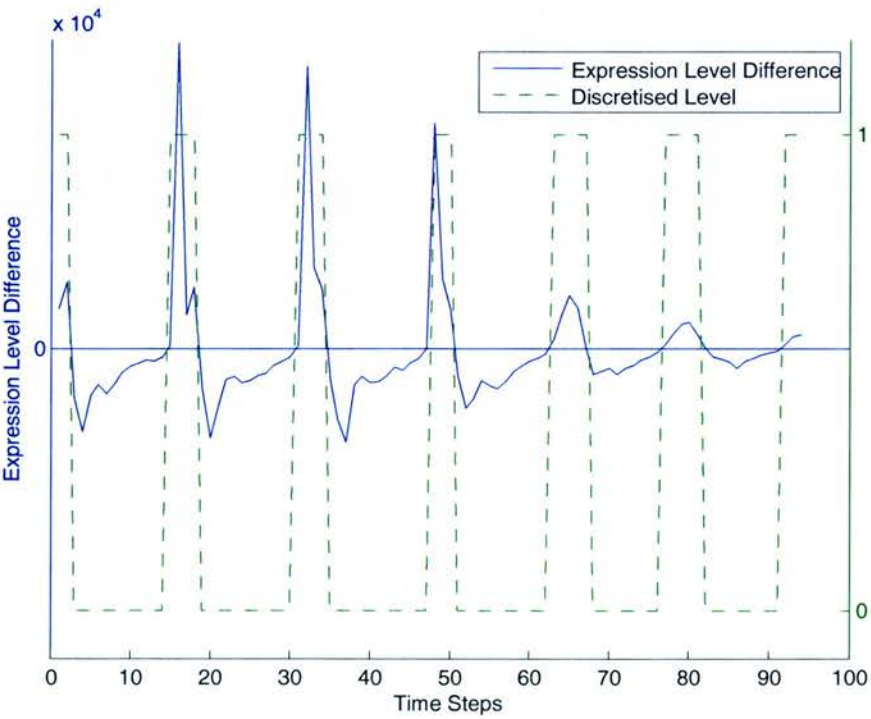


Figure 7.5: Expression level derivative of PRR9 in the AT0029 condition

oscillations present in the original data. To the author's knowledge, this is the first time this has been done when learning gene regulatory networks.

Finally due to the expert being able to provide knowledge on the behaviour of five of the genes and the light source, these were selected as the variables that would be used in the construction of the DBN. The genes in question were *GI*, *CCA1*, *LHY*, *PRR9* and *TOC1*.

7.3.2 Prior Knowledge

In order to provide more meaningful results and to compensate for the absence of much data, prior knowledge was incorporated into the learning algorithm in the form of constraints as to the allowable edges. These constraints are summarised below:

- Connections from a variable to the same variable at a later time point were disallowed. This type of information is trivial and does not add anything interesting to the model.
- No connection was allowed to the light variable, as this was causally independent of the gene expression values.
- Since interactions at this genetic level took time in the region of at least one time step, no connections were allowed among the variables at time t .
- A dynamic constraint was introduced that specified that if a connection was made from a variable X at time $t - i$ to a variable Y at time t , then no other connection could be made from X to Y . This is equivalent to saying that variables only affect each other temporally in a single manner, i.e. from X to Y only ever takes a set amount of time. Without a constraint such as this, in the absence of much data, multiple interactions between the same genes at different time points could easily be inferred.
- The number of slices of a possible dynamic Bayesian network was set to 9, i.e. the time slice at t and 8 slices back. Xing and Wu (2006) show using DBNs with multiple layers to model gene regulatory networks with multiple time lags. However, their methods consider arcs between arbitrary layers that are not at time t . This method is incorrect, as it means that arcs could be added between nodes without taking into account the other parents these nodes have.

7.3.3 Testing Methodology

In order to test the performance of ACO-E in recreating the expert's knowledge, a series of experiments were conducted. Each experiment used the data described in the preprocessing section, with the prior knowledge as described above. Also, the BDeu scoring criterion was used with an empty structure prior. The parameters of ACO-E were set as $\rho = 0.3$, $q_0 = 0.8$ and $\beta = 1$. These were selected as reasonable values that should perform well in most cases.

For each condition described in Table 7.1, experiments were performed, varying the equivalent sample size N' over 15 values ranging from 0.0001 to 500. These values were 0.0001, 0.001, 0.01, 0.1, 1, 3, 7, 10, 20, 30, 50, 80, 100, 300, 500. At each level of N' , 10 experiments were performed. Also, the data from all the conditions was concatenated and experiments performed, varying N' over 20 values ranging from 0.01 to 8000. These values were 0.01, 0.1, 1, 3, 7, 10, 20, 30, 50, 80, 100, 300, 500, 750, 1000, 2000, 4000, 6000, 7500, 8 000. Again, at each level of N' , 10 experiments were performed.

For each experiment, the resulting DBN was saved for later comparison against the expert supplied knowledge.

7.3.4 Evaluation Criteria

Evaluating the learnt DBNs was achieved by comparing them against the expert supplied knowledge. For this task, the knowledge was transformed into a DBN representing the state of the expert's knowledge on the domain. The DBN was validated by the expert and is shown in Figure 7.6. Note that nodes are not shown if they have no connection to any other node. Also, although the arcs in the network are definite, the knowledge of the domain expert was not as definite. Often the time points supplied were of the order of six hours long, i.e. instead of a definite time lag being given, the lag was bounded by two times roughly six hours apart. The effect of this would be to reduce the measured accuracy of any learned network.

To compare a learned network against the standard network, two measures were used: the true positive rate (TPR) and the false positive rate (FPR). These are defined as

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

and

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$

where

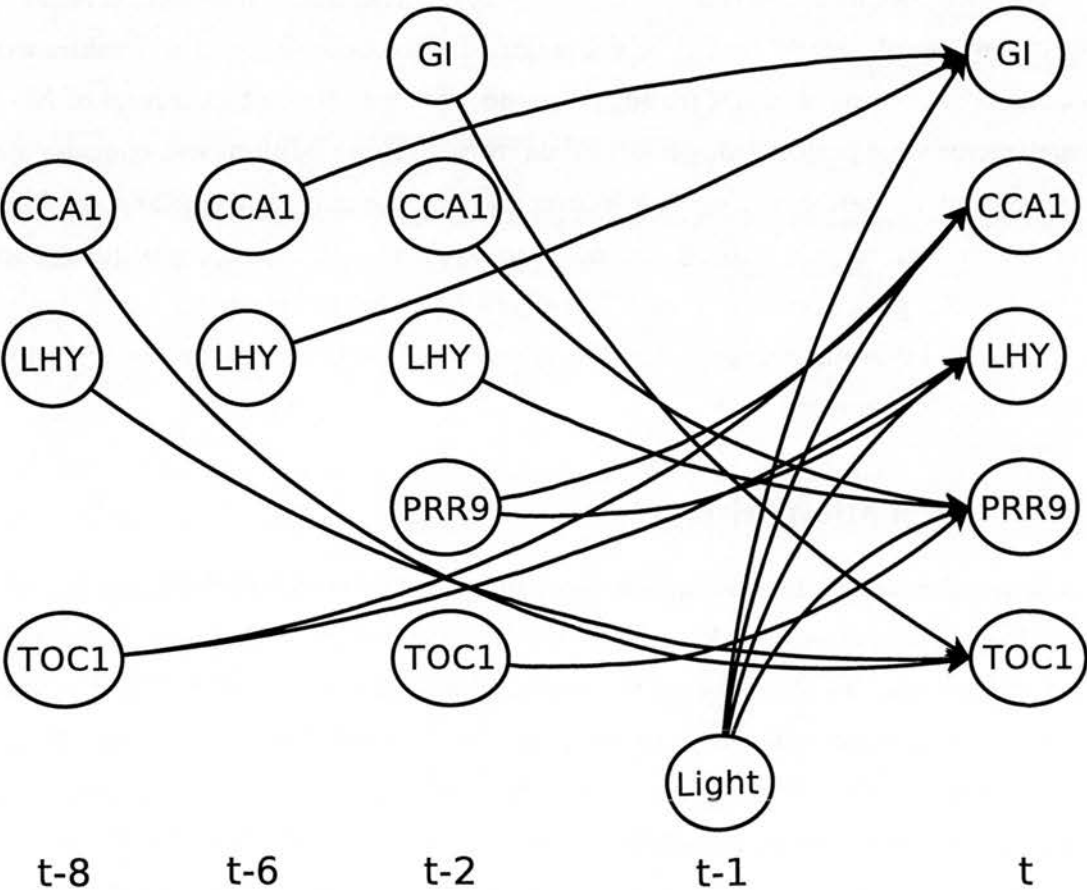


Figure 7.6: Expert developed DBN for *Arabidopsis thaliana* circadian clock

TP is the number of true positives, i.e. the number of arcs computed present, that are present. For the evaluation of the networks, this was calculated in two different ways:

Time Independent True Positive If there was an arc from gene X to gene Y in the standard network and also an arc from gene X to gene Y in the testing network, *irrespective of the time slice that X is in*, then this counts as a true positive.

Time Dependent True Positive If there was an arc from gene X to gene Y in the standard network and also an arc from gene X to gene Y in the testing network, then this counts as a *time dependent* true positive. The degree to which it is a true positive depends on the difference of time slices that the tail of each arc comes from. A difference of 0 gives a true positive of 1 and a difference of 8 gives a true positive of 0. Other values are calculated linearly between these two values. E.g. with an arc in the standard network having the tail in slice 4 and an arc in the testing network having a tail in slice 2, this leads to a difference of 2, which gives a time dependent true positive of $1 - 2/8 = 6/8$.

FP is the number of false positives, i.e. the number of arcs computed present that are absent. If there exists an arc from gene X to gene Y in the testing network and there does not exist an arc in the standard network, then this counts as a false positive.

TN is the number of true negatives, i.e. the number of arcs computed absent that are absent. If there does not exist an arc from gene X to gene Y in the testing network and there does not exist an arc in the standard network, then this counts as a true negative.

FN is the number of false negatives, i.e. the number of arcs computed absent that are present. If there does not exist an arc from gene X to gene Y in the testing network and there does exist an arc in the standard network, then this counts as a false negative.

In the work of Xing and Wu (2006), evaluating the learned network proceeds in a different manner than above. Some problems in their methods include:

Condition	Time Independent AUC	Time Dependent AUC	Light Ratio
All	0.7913	0.6749	—
AT0029	0.6943	0.5749	6:18
AT0030	0.6597	0.5942	9:15
AT0031b	0.8234	0.7046	12:12
AT0032	0.7135	0.5696	15:9
AT0033	0.8156	0.7155	18:6
AT0047	0.5200	0.4326	3:21

Table 7.3: Area under the ROC curve for both the time independent and time dependent true positive cases

- Firstly, as mentioned in Section 7.3.2, the method they use to add arcs can lead to incorrect conclusions about the structure of the dynamic network;
- Secondly, they don't take into account the time delay when evaluating the learned network; and
- Thirdly, they don't take any false positive rate into account.

7.4 Results and Discussion

For each condition and each level of N' , the mean of the TPR and that of the FPR were taken over the 10 experiments. This was done for the Time Independent TP (TITP) and the Time Dependent TP (TDTP). With these results, two different types of graphs were plotted. The first was the receiver operating characteristic (ROC) curve, which plots the FPR against the TPR. The second plots the FPR and TPR as a function of N' . The results of these plots are shown in Figures 7.7 to 7.16. Figures 7.7 to 7.10 show the results for the concatenated data, i.e. the data from all of the experimental conditions. Figures 7.11 to 7.16 show the results for each of the experimental conditions, from AT0029 to AT0047.

As well as the plots referred to above, the area under the curve (AUC) statistic is given in Table 7.3. The AUC is the area under the line of the ROC curve. It provides a single measure of the performance of an algorithm by integrating the FPR and TPR statistics. As a result of this, the distinction of the tradeoff between FPR and TPR is lost. However, it can sometimes be useful to have this measure as a means of comparing different algorithms.

7.4.1 Discussion of Experimental Results

The value of the equivalent sample size N' has a large effect on learning structure, as shown by Steck and Jaakkola (2002), Silander et al. (2007) and Kayaalp and Cooper (2002). Therefore, in learning a Bayesian network structure from data, it is important to see how this parameter will affect the learnt graph. This is the reason why the experiments described above were conducted over various values of N' .

Looking first at Figures 7.7 and 7.8, it can be seen that ACO-E does a good job of identifying all of the connections between the genes as supplied by the domain expert. In this case, the best value of N' is at 1, where the true positive rate is 1 and the false positive rate at just over 0.3. With the time dependent TPRs as seen in Figures 7.9 and 7.10, again the best value of N' is at 1, with a time dependent TPR of around 0.8 and a false positive rate of 0.3. In this case, 'best' is being taken as the absolute difference between the TPR and FPR.

These results show that ACO-E is efficient at finding all the connections between the genes, but not as proficient at keeping out bad connections. Indeed, when examining a trace of the algorithm it can be seen that spurious connections between highly synchronised genes are often inserted. E.g. if X causes Y and X causes Z , then a connection between Y and Z can easily appear. Problems such as these often appear with a small amount of data and prior knowledge becomes increasingly important in these situations.

With the results of the individual conditions, different behaviours can be observed. The value of N' for which best results are obtained differs widely depending on the condition used, ranging from 1 to 100. The different conditions also differ in how good the results are. E.g. the best performance came from the results with the larger light to dark ratio, i.e. AT0031b, AT0032 and AT0033. The worst performance came from the other conditions, i.e. AT0029, AT0030 and AT0047. Results such as these are plausible, as having more light in the entrainment phase equates to higher expression levels, which are less likely to be affected by noise.

It should be noted that not all the expert supplied knowledge is as accurate as may seem from the standard network. Whilst the time in that network was given in time steps of 1.5 hours, the knowledge of the expert was often given in terms of much larger grained steps (e.g. 'the morning'), as opposed to a definite time period. This is an artifact of the domain in question, as it is not completely understood at the present time. In turn, it has an effect on the time dependent true positive values obtained from comparing two networks; with better prior knowledge a more accurate comparison could be used that

bounds what the correct time lags might be for an arc.

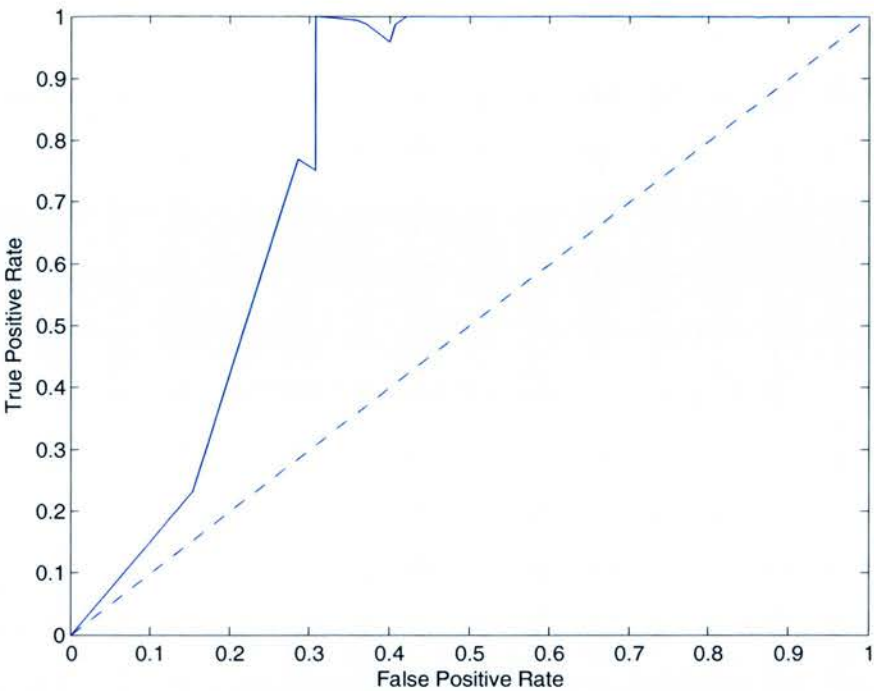


Figure 7.7: ROC for the time independent TPR and FPR for all data

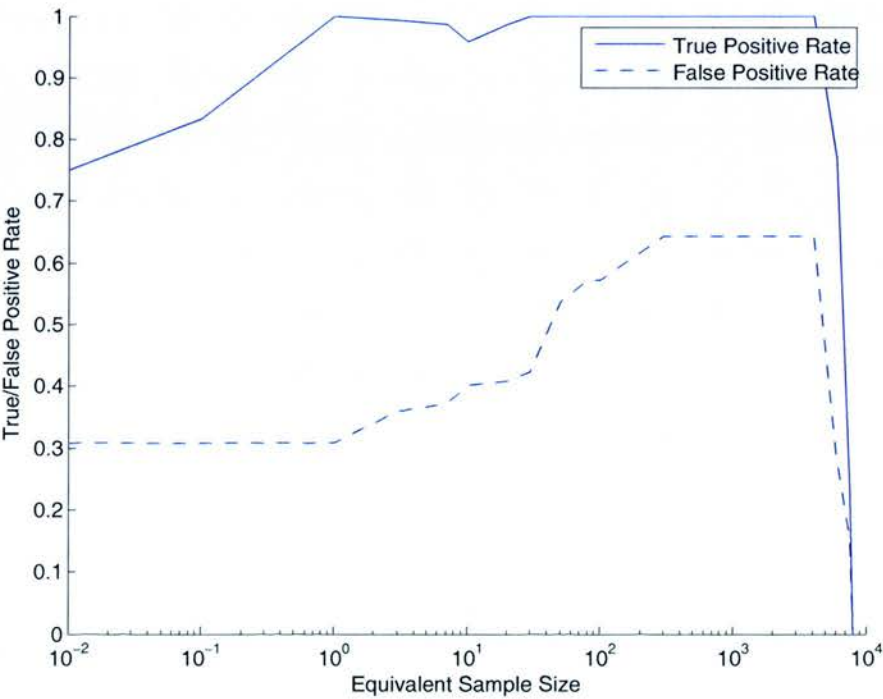


Figure 7.8: The time independent TPR and FPR as a function of N' for all data

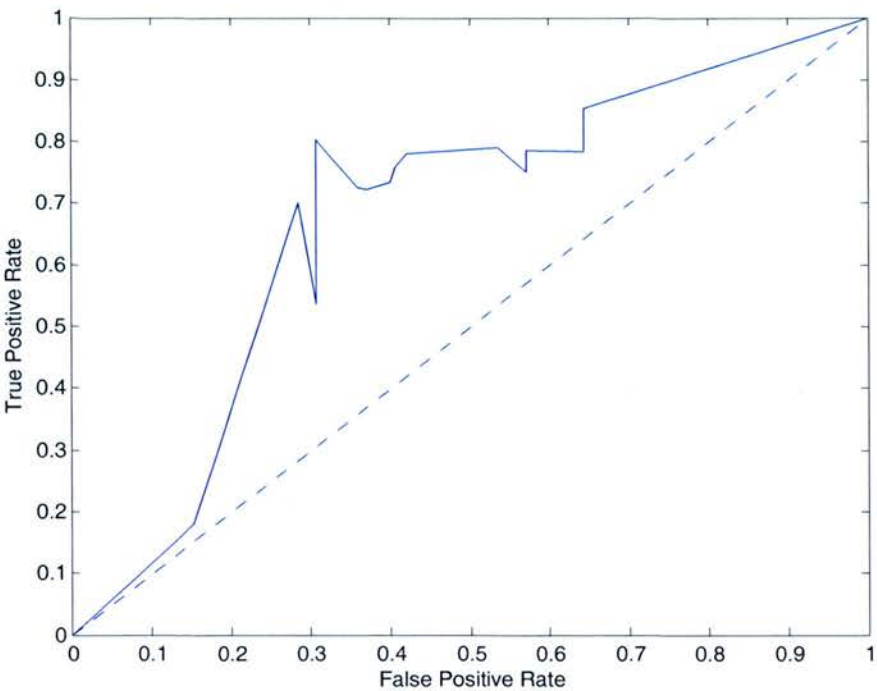


Figure 7.9: ROC for the time dependent TPR and FPR for all data

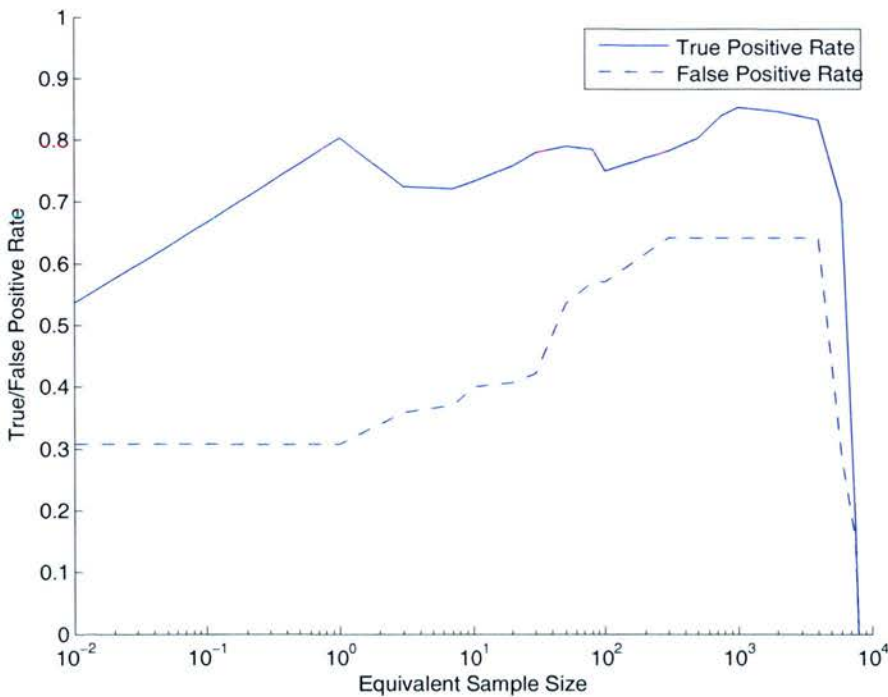


Figure 7.10: The time dependent TPR and FPR as a function of N' for all data

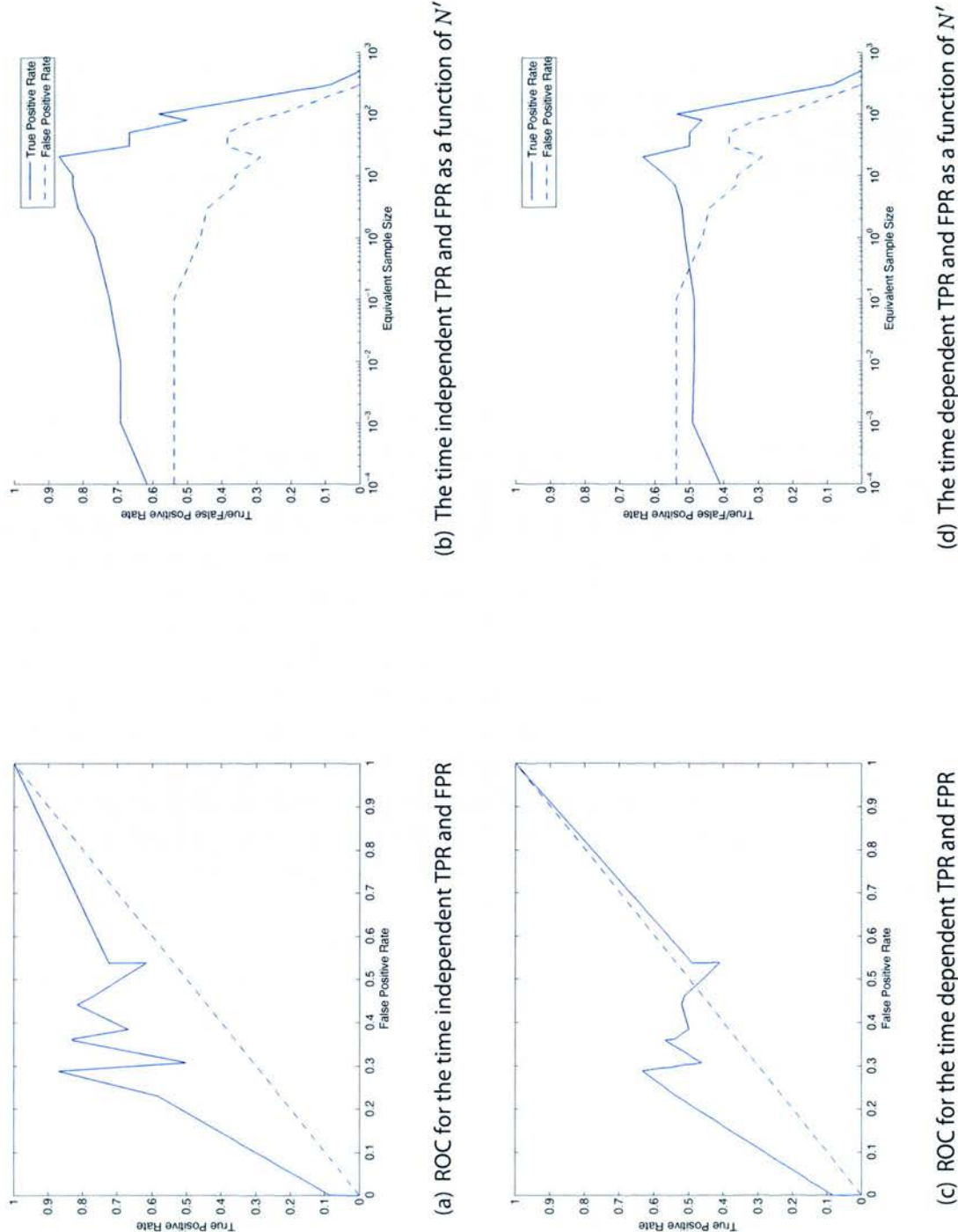


Figure 7.11: Plots of the positive rates for the AT0029 condition

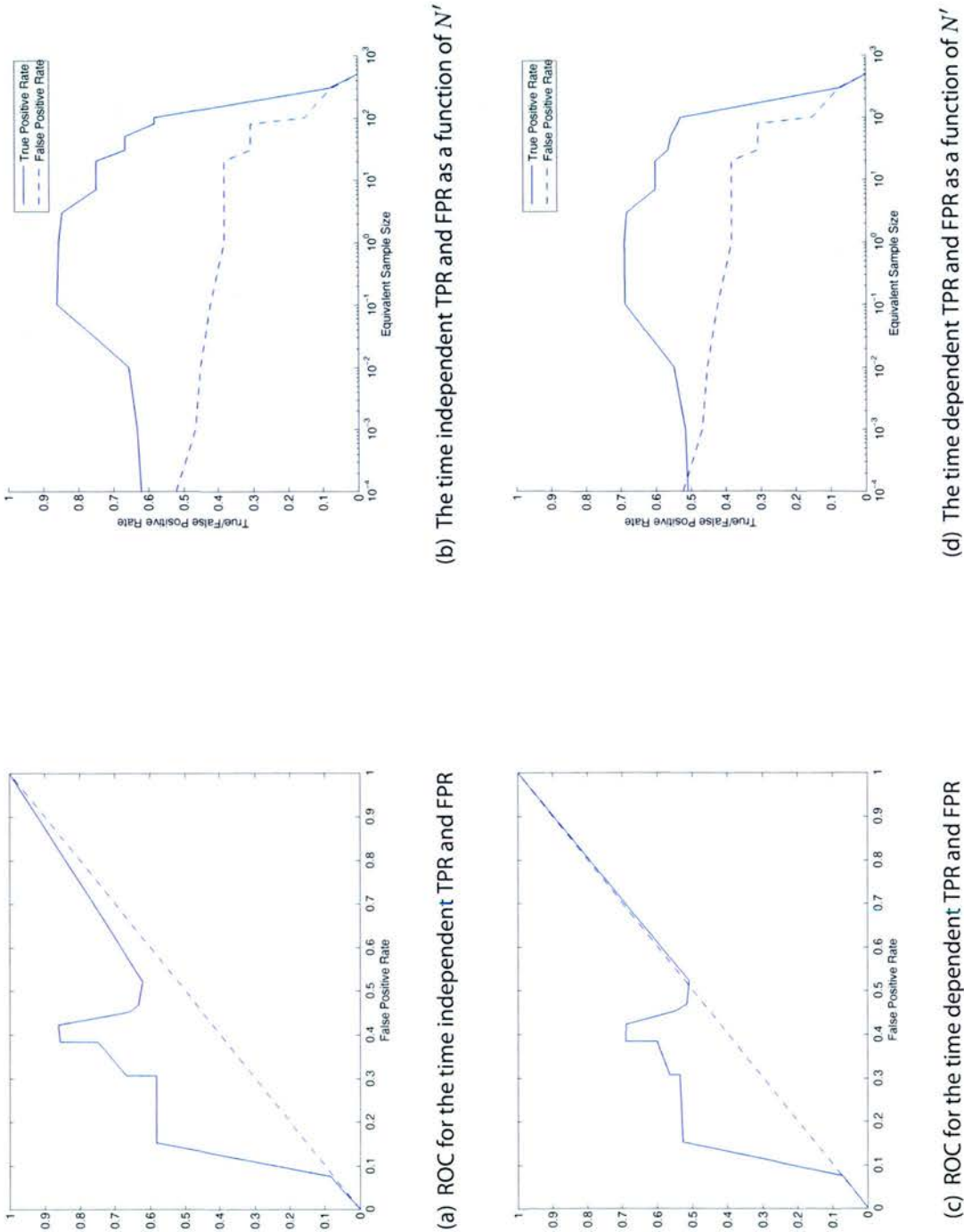


Figure 7.12: Plots of the positive rates for the AT0030 condition

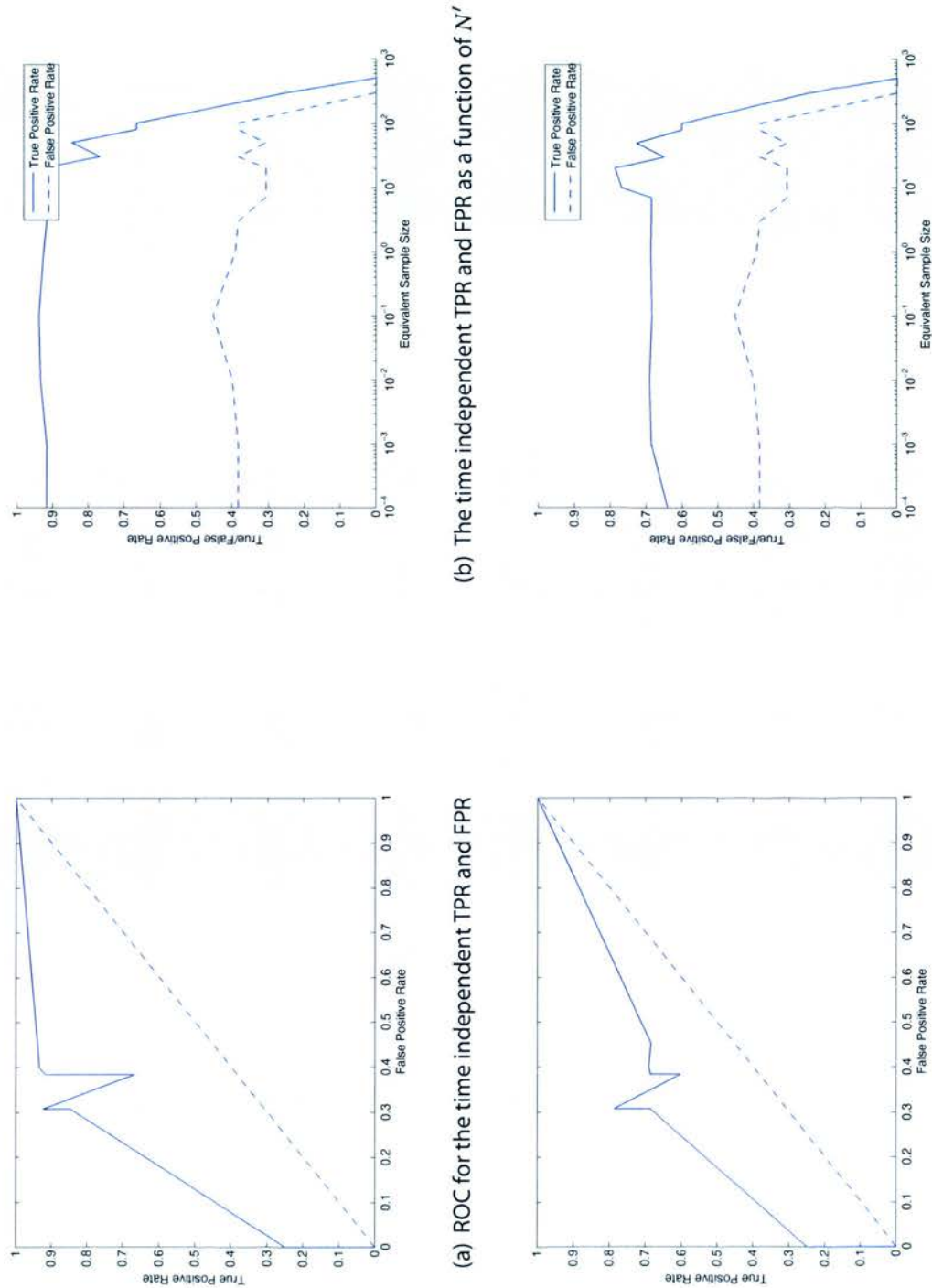


Figure 7.13: Plots of the positive rates for the AT0031b condition

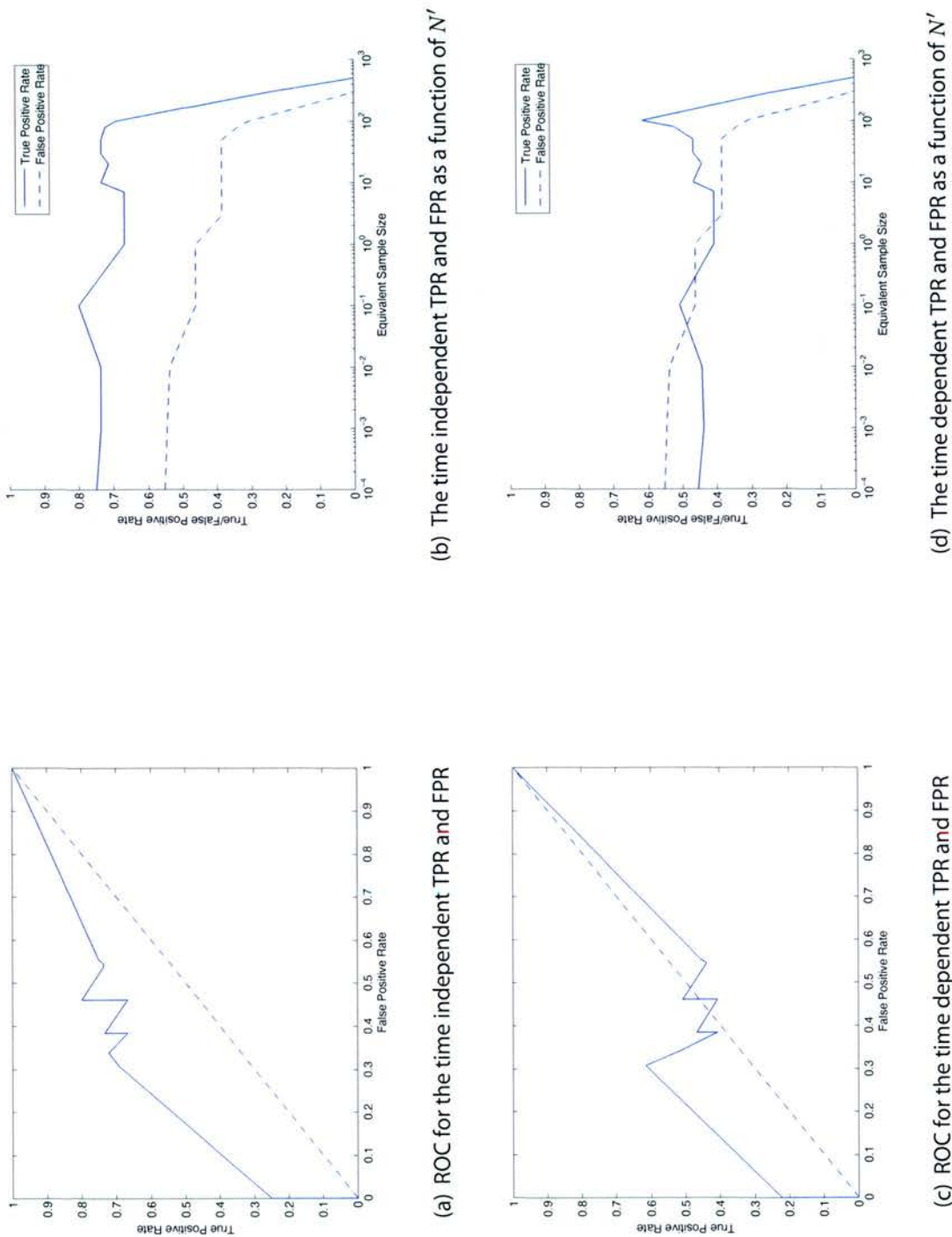


Figure 7.14: Plots of the positive rates for the AT0032 condition

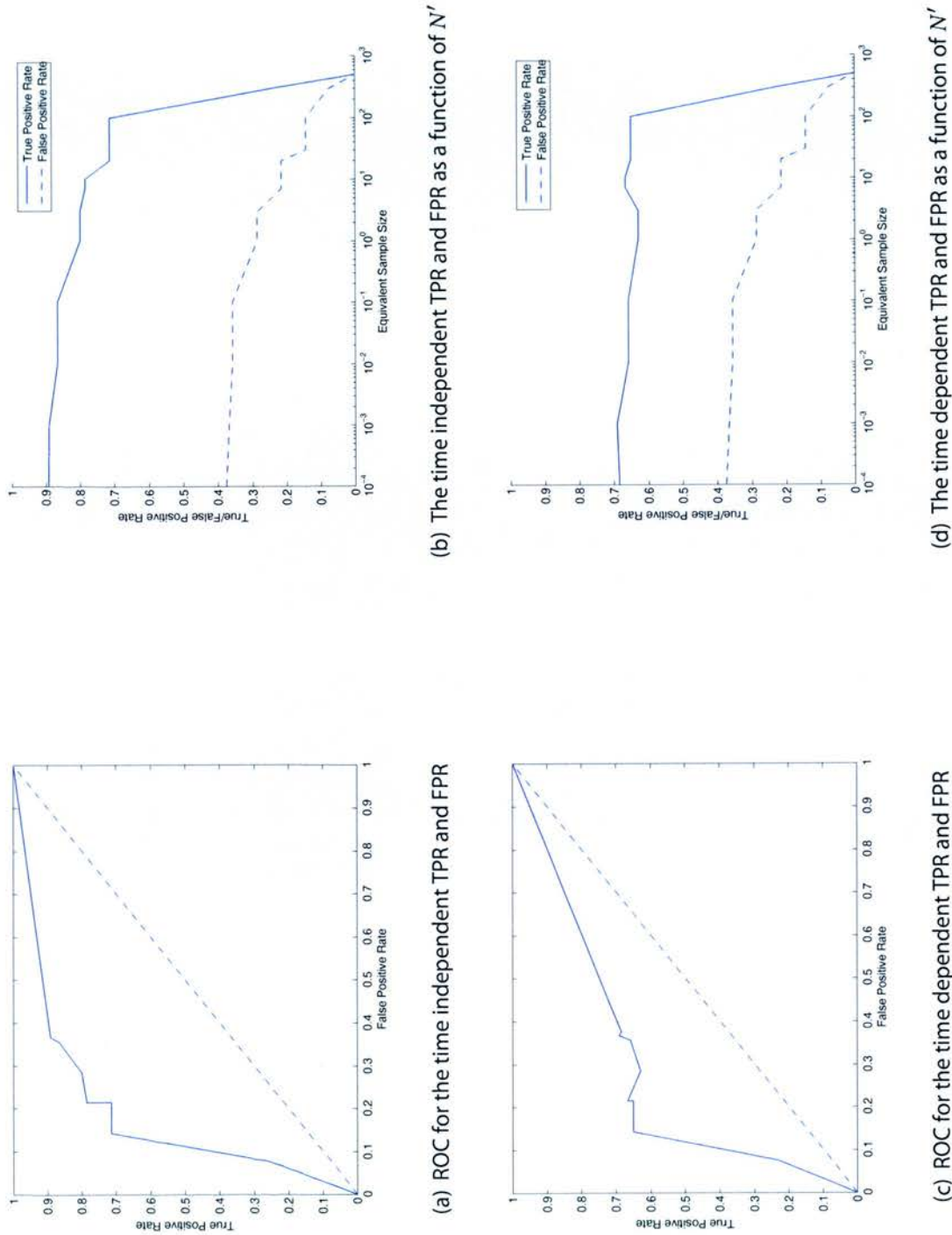
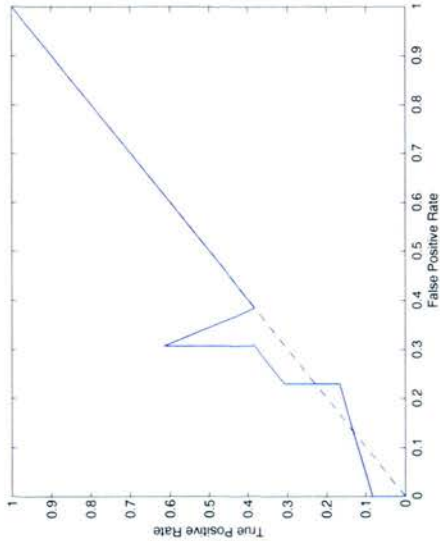
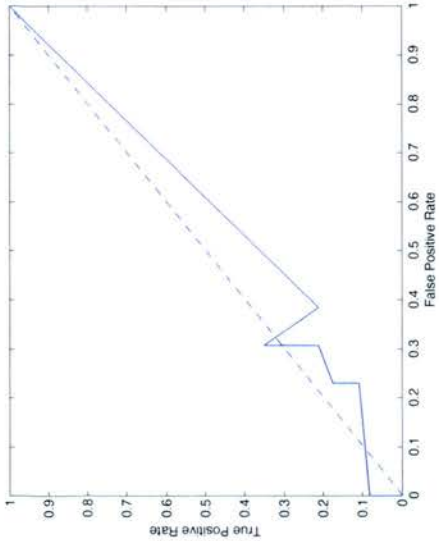


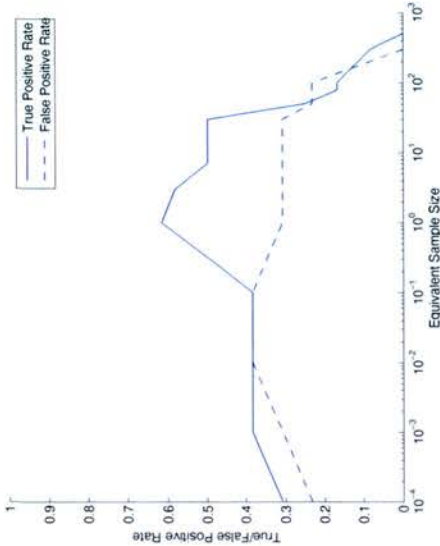
Figure 7.15: Plots of the positive rates for the AT0033 condition



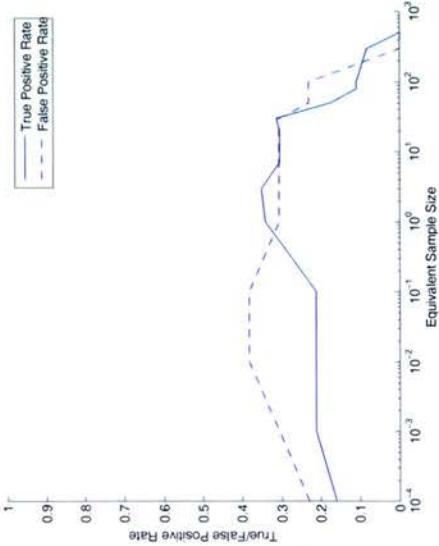
(a) ROC for the time independent TPR and FPR



(c) ROC for the time dependent TPR and FPR



(b) The time independent TPR and FPR as a function of N'



(d) The time dependent TPR and FPR as a function of N'

Figure 7.16: Plots of the positive rates for the AT0047 condition

7.5 Summary

This chapter developed an application of the ACO-E algorithm that involved learning gene regulatory networks from gene expression data. This was achieved by learning a dynamic Bayesian network. For this purpose the following tasks were completed.

Firstly, data from experiments involving the circadian clock of the *Arabidopsis thaliana* plant were obtained. The data involved six experimental conditions that varied the length of the day the plants perceived. This data was preprocessed by a novel method in relation to learning gene regulatory networks; the first difference was obtained and this difference discretised into two bins – rising or falling. Then the attributes that could be validated were selected and data to model the lighting conditions added.

Next the ACO-E algorithm was modified in order that valid dynamic Bayesian networks would be produced. This was achieved by making sure that arcs always went forward in time in the DBN. After this, prior knowledge was identified that would enable meaningful DBNs to be generated. The prior knowledge was needed because of the small amount of data available to the learning algorithm.

Because of the lack of much previous work on learning higher-order DBNs, new evaluation criteria had to be defined to judge how well the learned network reconstructed the expert supplied network. These criteria include:

- A method to provide a time dependent true positive rate depending on how close an arc is to the expert defined arc; and
- A consistent method to judge the true positives, true negatives, false positive and false negatives of a learned network compared to a standard network.

The algorithm was then tested over the different conditions and different equivalent sample size parameters. This was done in order to see whether the algorithm was under or over-estimating the amount of arcs that were added into a candidate network. Receiver operator characteristic (ROC) curves and plots of the true and false positive rates against N' were produced. These showed that the algorithm was adding in slightly too many arcs; the true positive rate was very high, with the false positive rate quite low. The area under the curve statistic was also shown for each of the conditions. This showed that the best results were for the conditions that had the most light in the entrainment phase of the experiment.

Chapter 8

Conclusions and Future Work

IN SUMMING UP the work of this dissertation, the motivation and objectives given in Chapter 1 should be recalled. Accordingly, the main points of these are given here. The motivations of this work were that:

- Bayesian networks are a useful tool to encode knowledge, but can be difficult to specify manually;
- Machine learning techniques are advantageous in building Bayesian networks, but often fall into problems with local maxima; and
- Ant colony optimisation is a useful tool in solving optimisation problems, that can avoid local maxima in a search space.

With these motivations, the broad aims of this thesis were to:

- Investigate strategies for building Bayesian network structures using machine learning techniques;
- Seek out and implement solutions to these problems;
- Test these solutions, against themselves with different parameter settings and against other competing techniques;
- Analyse the results of these tests in order to characterise the performance of the algorithms; and
- Test the solutions on a real-life problem in order to determine their effectiveness in an uncontrolled domain.

8.1 Conclusions

Although the main work of this thesis is on learning Bayesian network structures using ant colony optimisation, the aims given in the introduction and summarised above are more generic. In addition, they also specify extra objectives to be met. Therefore, the summary of the work done will be broken into three sections: speeding up searching through the space of equivalence classes of Bayesian networks, the ACO-E algorithm and the application of the ACO-E algorithm to data gathered from experiments performed on *Arabidopsis Thaliana*.

8.1.1 Accelerating the Search Through the Space of Equivalence Classes of Bayesian Network Structures

It was the comparative slowness of the ACO-E algorithm that motivated the techniques developed in Section 4.2. These methods were designed to speed up the run time of ACO-E, but are generic enough that they can be utilised on any technique that searches through the space of equivalence classes of Bayesian networks.

The results of Section 6.1 show that there is a significant difference between the run time of the ACO-E algorithm when the methods of Section 4.2 are used and when they are not used. These differences were found by running statistical tests across different gold-standard networks.

Furthermore, it was found that the speed-up was given as a function of the square root of the original running time. This was verified both empirically and analytically. In other words, the speed-up was dependent on the number of variables in the structure; the more variables, the greater the speed-up.

Finally it was found that any speed-up gained was the same for all the iterations of the algorithm, i.e. that it was invariant to the length of time that the algorithm was run.

8.1.2 Using Ant Colony Optimisation in Learning Equivalence Classes of Bayesian Network Structures

The main results in this thesis were on the development of the ACO-E algorithm as an implementation of the ACO metaheuristic to the problem of learning a Bayesian network structure that provides a good fit to a set of data. In a nutshell, ACO-E performed well in reconstructing a test network, from which data was sampled. A more detailed look

at the behaviour of ACO-E depending on its parameters, the type of test network and compared to other algorithms will now be given.

8.1.2.1 ACO-E Behaviour as its Parameters are Varied

In analysing the behaviour of ACO-E as a function of its parameters, the best and worst performing figures were compared, across each range of parameter. The best result was found when the parameter setting produced either the highest score or the smallest difference from the test network. The worst result was found when the parameter was 'switched off', i.e. when it had no effect on the algorithm's behaviour.

For all parameters, there was a difference between the behaviour of the best and the worst settings. Whether this difference was significant or not depended on the particular network being used as a test; some networks responded better to the algorithm than others. For those networks that ACO-E worked well with, the following trends were noticed:

- For data with more features, lower values of ρ , higher values of q_0 and higher values of β worked better; and
- For data with less features, higher values of ρ , lower values of q_0 and lower values of β worked better;

where ρ is the rate of pheromone deposition/evaporation, q_0 is the balance between exploration and exploitation and β is the power of the heuristic in the probabilistic transition rule.

8.1.2.2 The Utility of ACO-E as a Function of the Test Network

It was noticed that ACO-E performed better on some of the test networks than others. The networks that it fared best with were Barley, Mildew and Win95pts, described in Section 5.1.1. On closer examination of these networks it was found that they had nodes with a higher distribution of large in-degrees. This corresponded to a large average number of v-structures (as discussed in Section 2.1.3) per node value.

The reason that this might make a difference is because nodes with a large number of v-structures imply more possible local maxima in the search space. Greedy methods would run into these maxima, whereas ACO-E is able to find its way around them because of its stochastic nature of not always choosing the best move.

8.1.2.3 ACO-E Performance Compared to Similar Algorithms

The results of Section 6.2.2.6 show that ACO-E performs well against other algorithms that are similar in nature. These other algorithms were:

- GREEDY-E, which performs a greedy search in the space of equivalence classes of Bayesian network structures (Chickering, 2002a);
- EPQ, which performs an evolutionary programming search in the space of equivalence classes (Cotta and Muruzábal, 2004; Muruzábal and Cotta, 2004); and
- ACO-B, which performs a search using ACO in the space of DAGs (de Campos et al., 2002a).

In all cases, the BDeu score of ACO-E was better than the score of the other algorithms, at every iteration. In the case of the structural differences, it was better in all cases, except that of the HailFinder network, where the odd behaviour of the scoring function meant better BDeu scores implied worse structural differences. Concurring with the discussion above in 8.1.2.2, the networks for which ACO-E performed best were the Barley, Mildew and Win95pts networks.

8.1.2.4 ACO-E Performance Compared to Alternative State-of-the-Art Algorithms

Similar to the section above, ACO-E performed well in comparison to other state-of-the-art Bayesian network structure learning algorithms, performing better in 3 out the 4 tested: Barley, Mildew and HailFinder. The first two are networks in which it performed well in the self test. With the HailFinder network it is postulated that the results are good because of the search space; good results were also shown for the greedy equivalent search (GES) algorithm, which also searches through the space of equivalence classes.

Whilst ACO-E did not perform best with the Alarm network, it did not perform badly either, coming joint third in the rankings. The reasons for the performance on the Alarm network are discussed in Section 6.2.2.5.

8.1.3 Building a Dynamic Bayesian Network using ACO-E to Model the Circadian Clock of *Arabidopsis Thaliana*

Looking at the results of Section 7.4, it can be seen that ACO-E performed well in reconstructing the gene regulatory network of the circadian clock of the *Arabidopsis*

Thaliana plant. With these results there were two main points of interest:

- The equivalent sample size parameter N' of the BDeu scoring function played a large part in the accuracy of the reconstruction. In most cases, reasonable values of N' in the range $[1, 100]$ provided the best results. However, there was a tradeoff in the true positive rate and false positive rate as N' was varied.
- Spurious connections can easily be made between genes because of the hardness of detecting the correct parent given multiple possible time lags. E.g. if in reality, X causes Y with a lag of 3 and X causes Z with a lag of 8, then the data could also support X causing Y with a lag of 3 and Y causing Z with a lag of 5. Another example is if two genes happen to be synchronised, even though there is no relation between them. In this case, there is a high chance that a link will be placed between them.
- Because of the high chance of spurious connections, prior knowledge is very important in these situations, especially with small sample sizes. This knowledge can help constrain the possible states that a Bayesian network structure can take and so provide more meaningful results.
- The results are not conclusive because ACO-E was not compared against other algorithms with this data set.

8.2 Future Work

In performing the work in this thesis, there were many occasions that ideas had to be left untested due to time constraints. In this section it is hoped to note some of these, so that the current work can be built upon. ACO-E has shown good potential in accuracy of learning and there remain directions to be explored that could further improve this accuracy.

For the purposes of this section, any possible future work will be divided into three sections: the slowness of the ACO-E algorithm, extending the ACO-E algorithm so as to produce better results and the application of ACO-E to real-life data.

8.2.1 Speeding up ACO-E and Similar Algorithms

Because of the relative slowness of ACO-E in running, techniques were developed in Section 4.2 to speed this up. However, although these methods sped up the running time

tremendously, validity checking still remains the bottleneck when running ACO-E. Also, the methods are rather cumbersome to implement, especially in terms of updating the validity cache.

The main problem with these validity checks are the ‘path’ conditions that must be satisfied. Each of these requires a traversal of the graph that takes time in $O(v + e)$ at the very least, where v is the number of vertices and e is the number of edges. A better solution would be to have a single traversal of the graph find out all possible moves. For this to work, the properties of CPDAGs (as discussed in Section 2.1.3) would have to be examined, and a characterisation given of the types of moves allowed based on distance between vertices. Some of these characteristics have already been looked at with Section 4.2.1.

8.2.2 Extending ACO-E to Increase Performance and Scalability

Since validity checking is the slowest part of the ACO-E algorithm, it currently remains the first issue which must be dealt with, in order to improve running times. However, if that problem is solved then the focus will turn back to the other parts of the algorithm, particularly the scoring function.

Reducing the Number of Scoring Function Evaluations One very easy way in cutting down the number of score evaluations would be to have a static heuristic defined that could say, e.g. what would be the benefit of adding an arc to the empty graph. In this way, scoring functions would only have to be evaluated once per move and hence lead to a speeding up of the algorithm. With a situation like this, local search would become more important in order to ‘finish off’ traversals to the best possible positions.

Comparing the Computational Complexity of ACO-E to Other Algorithms

Scoring function evaluations are often used as a measure of complexity in Bayesian network structure learning algorithms. However, this quantity was not measured for experiments on ACO-E. In the future, a proper comparison of this figure to those of other structure learning algorithms would enable the complexity of ACO-E to be quantified. This is useful as it would provide a measure of the scalability of ACO-E to situations with large numbers of variables.

Pruning the Search Space Recently, hybrid learning algorithms, as discussed in Section 2.4.8, have shown good success in learning Bayesian network structures, whilst

cutting down on running time, sometimes dramatically. They generally work by using a conditional independence test to discover nodes that would likely be connected to a given node and remove the rest of them from consideration. This has the effect of requiring less scoring function evaluations, thus speeding up the algorithm and requiring less memory to store the results of evaluations. With no bound on the number of possible parents, the number of cached values would grow at least quadratically with the number of variables and eventually exhaust the computer's memory.

Applying ACO-E with Different Search Operators to Better Avoid Local Maxima

According to Castelo and Kočka (2003), there are certain operators that are able to avoid local maxima in a search space, provided that the sample size tends to infinity. An example of these are the operators given by Chickering (2002b) that are used in a greedy search in the space of equivalence classes of structures (GES).

However, at small sample sizes these guarantees are not strictly true and search algorithms can still get caught in maxima. An example of a method that tries to avoid these is the KES algorithm of Nielsen et al. (2003) that uses the operators in GES, but has a parameter that controls how often the algorithm acts greedily; when the algorithm does not act greedily, it chooses a move that is not necessarily the best. Experiments show that KES behaves better than GES most of the time.

This procedure bears some similarities to ACO-E. If the randomness was augmented by pheromone and heuristics, there is a possibility that performance would improve even more.

8.2.3 Applying ACO-E to Real-Life Data

In real-life situations, it is not normal to apply a learning method to a set of data without sanitising it in some way, especially when the size of the sample is small. When applying ACO-E to the *Arabidopsis Thaliana* data, the only preprocessing that was done to the data was to take the first difference and discretise it.

When the raw data and this difference was analysed, it was found to contain noise, particularly in the conditions where there was not much light in the entrainment phase. Without enough light, certain genes were not able to start expressing themselves and get in phase with the clock. Hence the signal to noise ratio was quite low and artifacts crept into the discretised data, with spikes appearing at the areas with a low expression level. This noise would generally have the property of making the results of the various tests worse.

In order to handle this, some filtering could be used to clear up the signal. A low-width median filter applied to the discretised domain would remove the spikes; more troublesome noise could use a wider-band filter. Applying these techniques in consultation with a domain expert would ensure that minimal information was discarded.

Also, comparing ACO-E against other Bayesian network structure learning techniques on this data would better characterise its performance and provide a measure of the problem difficulty.

8.3 Summary

This thesis was concerned with using ACO in learning Bayesian network structures. In investigating this topic, two novel techniques were devised. The first new technique dealt with speeding up the learning of Bayesian network structures whilst searching through the space of equivalence classes of structures. This approach produced an order of magnitude speed-up in learning structures.

The second new technique devised was the ACO-E algorithm. This used ACO to conduct a search through the space of equivalence classes of Bayesian network structures. The algorithm performed better than many other state-of-the-art structure learning algorithms, over a range of performance measures, in six benchmark data sets. ACO-E was also able to partially reconstruct a gene regulatory network of *Arabidopsis Thaliana*. It did this by constructing an eighth-order dynamic Bayesian network, as opposed to the first-order networks that are normally created.

ACO-E is a practical algorithm, that has shown great promise in the field of Bayesian network structure learning. Whilst it performs well in many diverse problem domains, there are additional innovative directions to be explored, which may further improve on the results reported in this thesis.

Appendix A

Experimental Bayesian Networks

IN THE EXPERIMENTS shown in this thesis, six gold-standard networks are used. These are the Alarm (Beinlich et al., 1989), Barley (Kristensen and Rasmussen, 2002), Diabetes (Andreassen et al., 1991), HailFinder (Abramson et al., 1996), Mildew (Jensen, 1995) and Win95pts networks (Microsoft Research, 1995). These networks were chosen because they covered a wide range of domains, were easily available and all contained discrete attributes. The last property was important because the scoring criterion that would be used in the experiments is implemented over multinomial random variables.

Figures A.1 to A.6 show the structure of the Bayesian networks in question. Various properties of these Bayesian networks are shown in Table 5.1.

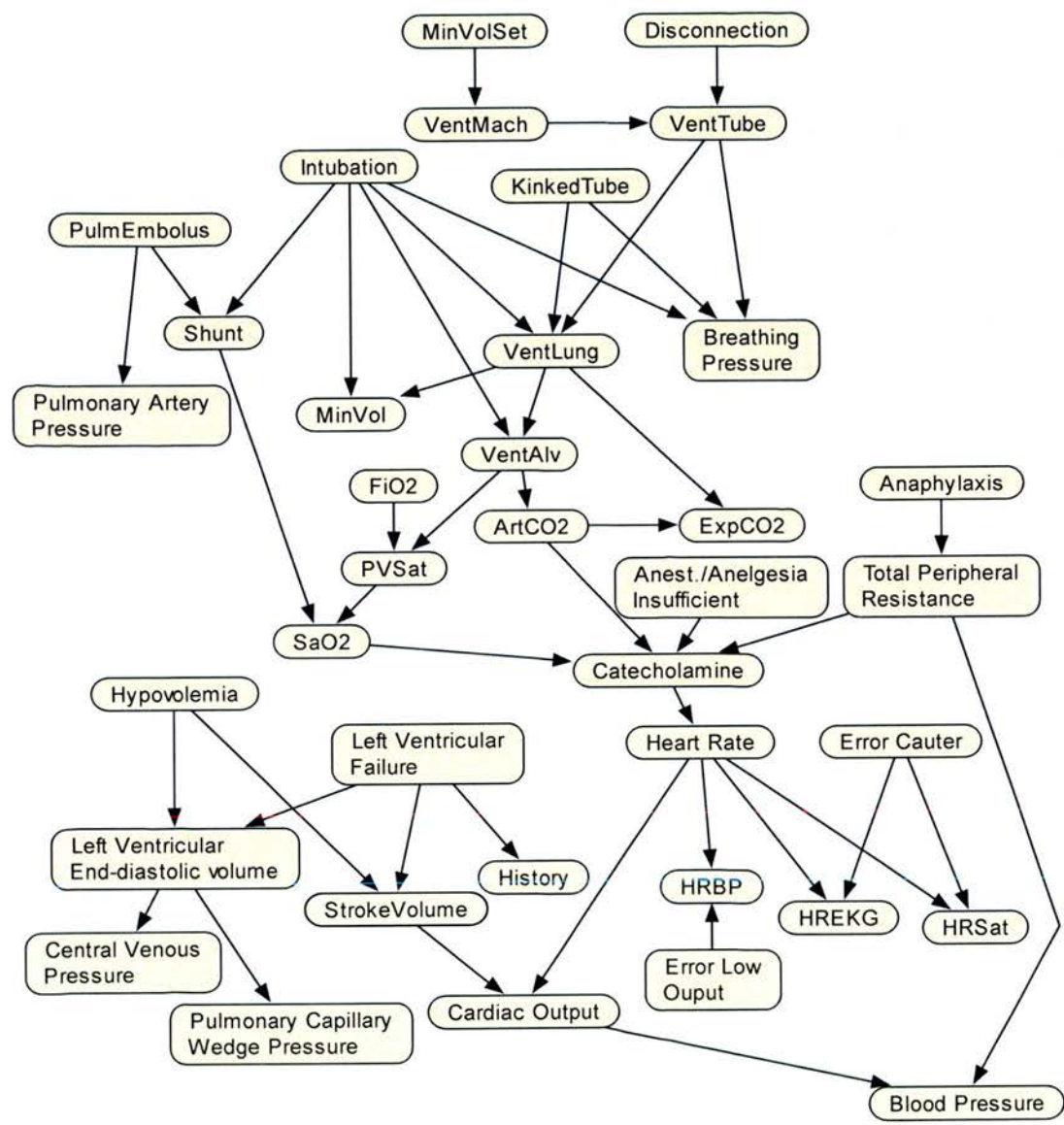


Figure A.1: The Alarm Bayesian network

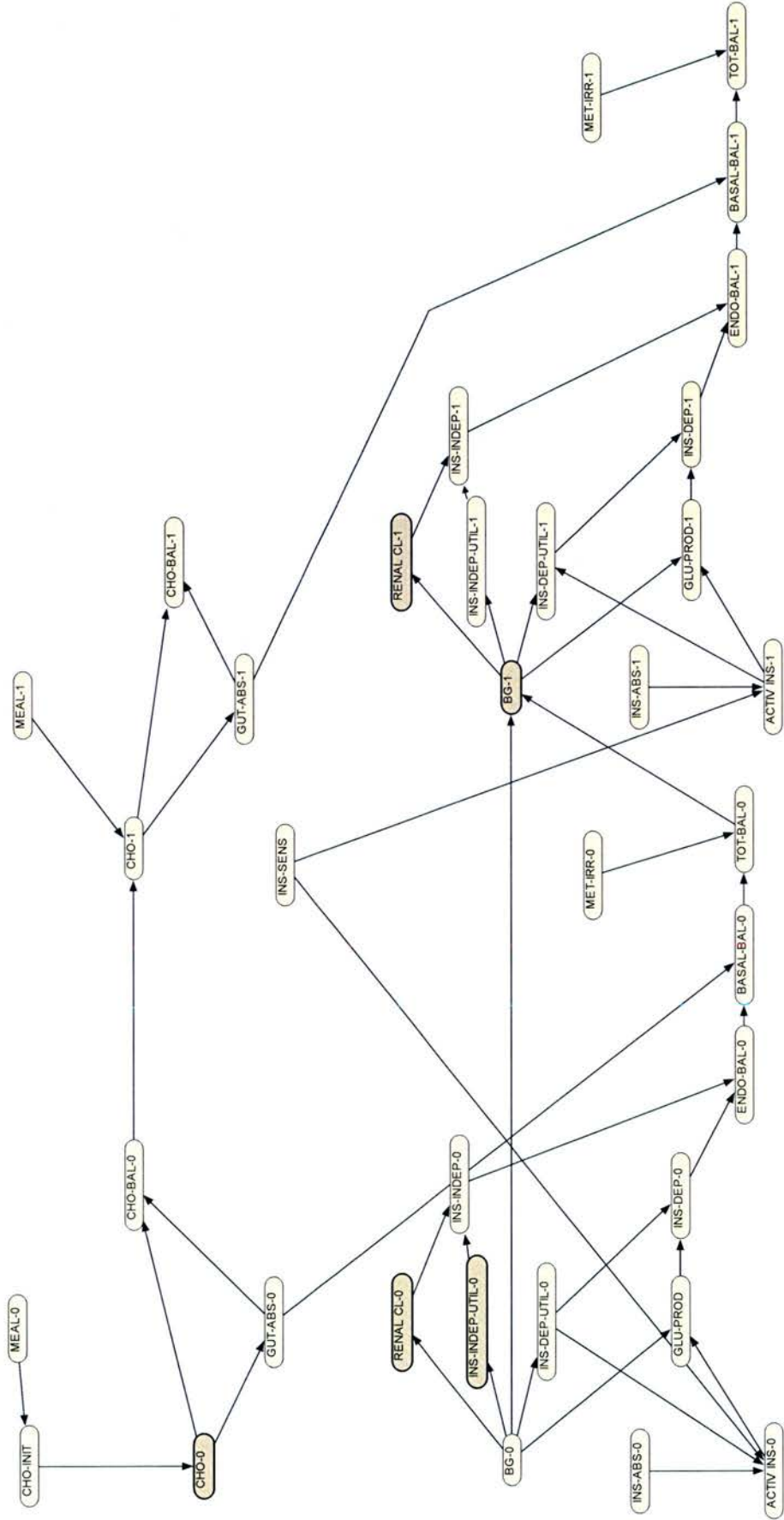


Figure A.3: The Diabetes Bayesian network

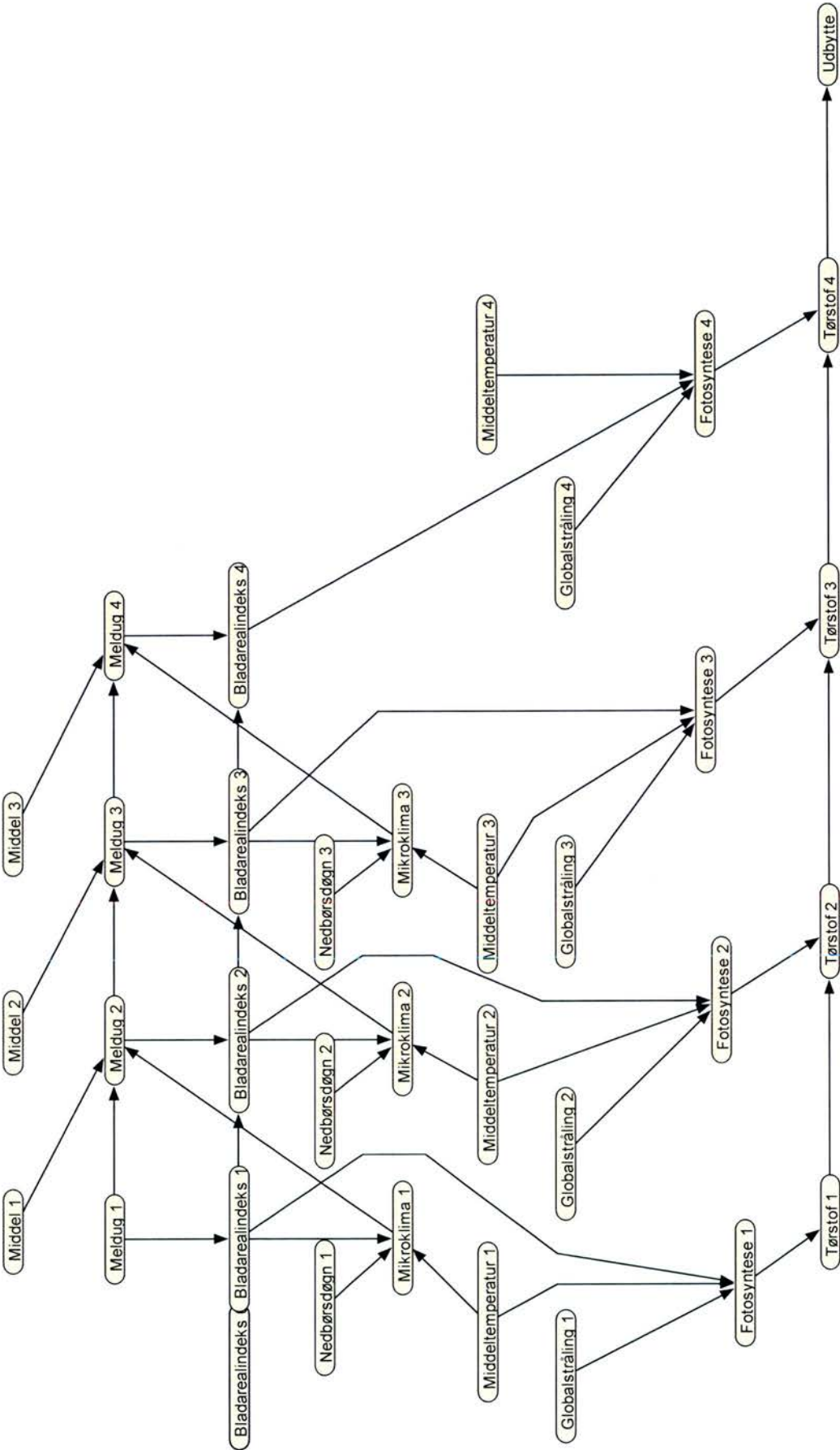


Figure A.5: The Mildew Bayesian network

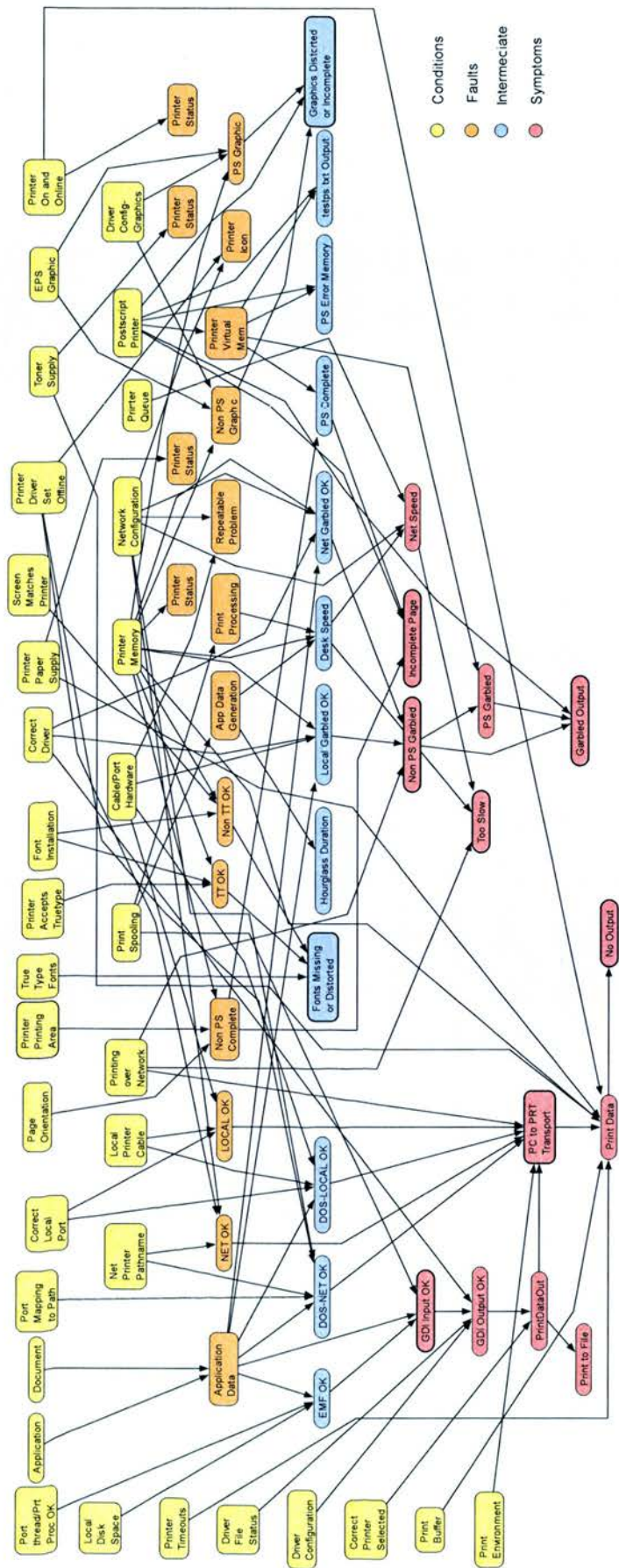


Figure A.6: The Win95pts Bayesian network

Bibliography

- Bruce Abramson, John Brown, Ward Edwards, Allan Murphy and Robert L. Winkler (1996). Hailfinder: A Bayesian system for forecasting severe weather. *International Journal of Forecasting*, 12(1):57–71. doi:10.1016/0169-2070(95)00664-8.
- Bruce Abramson and Anthony Finizza (1991). Using belief networks to forecast oil prices. *International Journal of Forecasting*, 7(3):299–315. doi:10.1016/0169-2070(91)90004-F.
- Silvia Acid and Luis M. De Campos (2000). Learning right sized belief networks by means of a hybrid methodology. In D.A. Zighed, J. Komorowski and J. Żytkow, editors, *Principles of Data Mining and Knowledge Discovery: 4th European Conference, PKDD 2000*, volume 1910 of *Lecture Notes in Artificial Intelligence*, pp. 309–315. Springer.
- Silvia Acid and Luis M. de Campos (1995). Approximations of causal networks by poly-trees: an empirical study. In *Advances in Intelligent Computing — IPMU '94*, volume 945 of *Lecture Notes in Computer Science*, pp. 149–158. Springer. doi:10.1007/BFb0035946.
- Silvia Acid and Luis M. de Campos (1996a). An algorithm for finding minimum d-separating sets in belief networks. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 3–10. Morgan Kaufmann.
- Silvia Acid and Luis M. de Campos (1996b). An algorithm for finding minimum d-separating sets in belief networks. Technical Report DECSAI-96-02-14, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada.
- Silvia Acid and Luis M. de Campos (1996c). BENEDICT: An algorithm for learning probabilistic Bayesian networks. In *Proceedings of the Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 979–984.

- Silvia Acid and Luis M. de Campos (2001). A hybrid methodology for learning belief networks: BENEDICT. *International Journal of Approximate Reasoning*, 27(3):235–262. doi:10.1016/S0888-613X(01)00041-X.
- Silvia Acid and Luis M. de Campos (2003). Searching for Bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18:445–490.
- Silvia Acid, Luis M. de Campos, Juan M. Fernandez-Luna, Susana Rodriguez, Jose Maria Rodriguez and Jose Luis Salcedo (2004). A comparison of learning algorithms for Bayesian networks: a case study based on data from an emergency medical service. *Artificial Intelligence in Medicine*, 30(3):215–232. doi:10.1016/j.artmed.2003.11.002.
- Silvia Acid, Luis M. de Campos and Juan F. Huete (2001). The search of causal orderings: A short cut for learning belief networks. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty: Proceedings of the Sixth European Conference, ECSQARU 2001*, volume 2143 of *Lecture Notes in Artificial Intelligence*, pp. 216–227. Springer.
- Hirotsugu Akaike (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723. doi:10.1109/TAC.1974.1100705.
- Constantin F. Aliferis and Ioannis Tsamardinos (2002). Algorithms for large-scale local causal discovery and feature selection in the presence of limited sample or large causal neighbourhoods. Technical Report DSL-02-08, Department of Biomedical Informatics, Vanderbilt University.
- Steen A. Andersson, David Madigan and Michael D. Perlman (1997). A characterization of markov equivalence classes for acyclic digraphs. *The Annals of Statistics*, 25(2):505–541. doi:10.1214/aos/1031833662.
- S. Andreassen, F. V. Jensen, S. K. Andersen, B. Falck, U. Kjøl, M. Woldbye, A. R. Sørensen, A. Rosenfalck and F. Jensen (1989). MUNIN—an expert EMG assistant. In *Computer-Aided Electromyography and Expert Systems*, pp. 255–277. Elsevier.
- Steen Andreassen, R. Hovorka, JJ Benn, Kristian G. Olesen and E. Carson (1991). A model-based approach to insulin adjustment. In *Proceedings of the Third Conference on Artificial Intelligence in Medicine, AIME '91*, pp. 239–249.

- Francis R. Bach and Michael I. Jordan (2003). Learning graphical models with Mercer kernels. In Suzanna Becker, Sebastian Thrun and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, pp. 1009–1016. MIT Press.
- Shumeet Baluja (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, School of Computer Science, Carnegie Mellon University.
- Eric Bauer, Daphne Koller and Yoram Singer (1997). Update rules for parameter estimation in Bayesian networks. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pp. 3–13. Morgan Kaufmann.
- Matthew J. Beal and Zoubin Ghahramani (2003). The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. In J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith and M. West, editors, *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*, pp. 453–464. Oxford University Press.
- Ann Becker and Dan Geiger (1994). Approximation algorithms for the loop cutset problem. In Ramon Lopez de Mantaras and David Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pp. 60–68. Morgan Kaufmann.
- Ann Becker and Dan Geiger (1996a). Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83(1):167–188. doi:10.1016/0004-3702(95)00004-6.
- Ann Becker and Dan Geiger (1996b). A sufficiently fast algorithm for finding close to optimal junction trees. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 81–89. Morgan Kaufmann.
- Ann Becker and Dan Geiger (2001). A sufficiently fast algorithm for finding close to optimal clique trees. *Artificial Intelligence*, 125(1–2):3–17. doi:10.1016/S0004-3702(00)00075-8.
- I. Beinlich, H. Suermondt, R. Chavez and G. Cooper (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks.

- In *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, pp. 247–256. Springer.
- Kristin P. Bennett and Emilio Parrado-Hernández (2006). The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, 7:1265–1281.
- Allister Bernard and Alexander J. Hartemink (2005). Informative structure priors: Joint learning of dynamic regulatory networks from multiple types of data. In Russ B. Altman, Tiffany A. Jung, Teri E. Klein, A. Keith Dunker and Lawrence Hunter, editors, *Proceedings of the Pacific Symposium on Biocomputing (PSB 2005)*, pp. 459–470. World Scientific.
- John Binder, Daphne Koller, Stuart Russell and Keiji Kanazawa (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2–3):213–244. doi:10.1023/A:1007421730016.
- Christopher Bishop, Neil Lawrence, Tommi Jaakkola and Michael Jordan (1998). Approximating posterior distributions in belief networks using mixtures. In Michael I. Jordan, Michael J. Kearns and Sara A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pp. 416–422. MIT Press.
- Rosa Blanco, Iñaki Inza and Pedro Larrañaga (2003). Learning Bayesian networks in the space of structures by estimation of distribution algorithms. *International Journal of Intelligent Systems*, 18(2):205–220. doi:10.1002/int.10084.
- Christian Blum and Krzysztof Socha (2005). Training feed-forward neural networks with ant colony optimization: An application to pattern classification. In Nadia Nedjah, Luiza de Macedo Mourelle, Marley Maria Bernardes Rebuszi Vellasco, Ajith Abraham and Mario Köppen, editors, *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems (HIS '05)*. IEEE. doi:10.1109/ICHIS.2005.104.
- Eric Bonabeau, Marco Dorigo and Guy Theraulaz (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Studies in the Sciences of Complexity. Oxford University Press.
- Hanen Borchani, Nahla Ben Amor and Khaled Mellouli (2006). Learning Bayesian network equivalence classes from incomplete data. In *Discovery Science: 9th International Conference*, volume 4265 of *Lecture Notes in Artificial Intelligence*, pp. 291–295. Springer. doi:10.1007/11893318_29.

- Remco R. Bouckaert (1993). Probabilistic network construction using the minimum description length principle. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty: European Conference ECSQARU '93*, volume 747 of *Lecture Notes in Computer Science*, pp. 41–48. Springer. doi:10.1007/BFb0028180.
- Remco R. Bouckaert (1994a). Probabilistic network construction using the minimum description length principle. Technical Report RUU-CS-94-27, Department of Computer Science, Utrecht University.
- Remco R. Bouckaert (1994b). Properties of measures for Bayesian belief network learning. Technical Report UU-CS-1994-35, Department of Information and Computing Sciences, Utrecht University.
- Remco R. Bouckaert (1994c). A stratified simulation scheme for inference in Bayesian belief networks. Technical Report UU-CS-1994-16, Department of Computer Science, Utrecht University.
- Remco R. Bouckaert, Enrique Castillo and José Manuel Gutiérrez (1996). A modified simulation scheme for inference in Bayesian networks. *International Journal of Approximate Reasoning*, 14(1):55–80. doi:10.1016/0888-613X(95)00114-V.
- Craig Boutilier, Nir Friedman, Moises Goldszmidt and Daphne Koller (1996). Context-specific independence in bayesian networks. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 115–123. Morgan Kaufmann.
- Xavier Boyen, Nir Friedman and Daphne Koller (1999). Discovering the hidden structure of complex dynamic systems. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 91–100. Morgan Kaufmann.
- Xavier Boyen and Daphne Koller (1998). Tractable inference for complex stochastic processes. In Gregory F. Cooper and Serafin Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 33–42. Morgan Kaufmann.
- Jack S. Breese and Eric Horvitz (1991). Ideal reformulation of belief networks. In Piero Bonissone, Max Henrion, Laveen Kanal and John Lemmer, editors, *Uncertainty in Artificial Intelligence 6*, pp. 129–144. North-Holland.

- Laura E. Brown, Ioannis Tsamardinos and Constantin F. Aliferis (2004). A novel algorithm for scalable and accurate Bayesian network learning. In *Proceedings of the Eleventh World Congress on Medical Informatics (MEDINFO)*, volume 1, pp. 711–715. IOS Press, Amsterdam.
- Laura E. Brown, Ioannis Tsamardinos and Constantin F. Aliferis (2005). A comparison of novel and state-of-the-art polynomial bayesian network learning algorithms. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings of the Twentieth National Conference On Artificial Intelligence*, volume 2, pp. 739–745. AAAI Press.
- Bernd Bullnheimer, Richard F. Hartl and Christine Strauss (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89:319–328. doi:10.1023/A:1018940026670.
- Wray Buntine (1991). Theory refinement on Bayesian networks. In Bruce D'Ambrosio, Philippe Smets and Piero Bonissone, editors, *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 52–60. Morgan Kaufmann, San Mateo, CA.
- Wray Buntine (1996). A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210. doi:10.1109/69.494161.
- Wray L. Buntine (1994). Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225.
- John Burge and Terran Lane (2006). Improving Bayesian network structure search with random variable aggregation hierarchies. In *Proceedings of the Seventeenth European Conference on Machine Learning (ECML 2006)*, volume 4212 of *Lecture Notes in Artificial Intelligence*, pp. 66–77. Springer. doi:10.1007/11871842_11.
- Susanne Gammelgaard Bøttcher (2004). *Learning Bayesian Networks with Mixed Variables*. Ph.D. thesis, Department of Mathematical Sciences, Aalborg University.
- Jose E. Cano, Luis D. Hernández and Serafin Moral (1996). Importance sampling algorithms for the propagation of probabilities in belief networks. *International Journal of Approximate Reasoning*, 15(1):77–92. doi:10.1016/0888-613X(96)00013-8.
- Robert Castelo and Tomáš Kočka (2003). On inclusion-driven learning of Bayesian networks. *Journal of Machine Learning Research*, 4:527–574.

- Robert Castelo and Michael D. Perlman (2002). Learning essential graph markov models from data. In *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM 2002)*, pp. 17–24.
- Robert Castelo and Arno Siebes (2000). Priors on network structures. Biasing the search for Bayesian networks. *International Journal of Approximate Reasoning*, 24(1):39–57. doi:10.1016/S0888-613X(99)00041-9.
- Enrique Castillo, José Manuel Gutiérrez and Ali S. Hadi (1997a). *Expert Systems and Probabilistic Network Models*. Monographs in Computer Science. Springer.
- Enrique Castillo, José Manuel Gutiérrez and Ali S. Hadi (1995). Parametric structure of probabilities in Bayesian networks. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU '95)*, volume 946 of *Lecture Notes in Artificial Intelligence*, pp. 89–98. Springer. doi:10.1007/3-540-60112-0_11.
- Enrique Castillo, José Manuel Gutiérrez and Ali S. Hadi (1996). A new method for efficient symbolic propagation in discrete bayesian networks. *Networks*, 28(1):31–43. doi:10.1002/(SICI)1097-0037(199608)28:1<31::AID-NET5>3.0.CO;2-E.
- Enrique Castillo, Ali S. Hadi and Cristina Solares (1997b). Learning and updating of uncertainty in Dirichlet models. *Machine Learning*, 26(1):43–63. doi:10.1023/A:1007372016040.
- Allen Chan and Alex Freitas (2006a). A new classification-rule pruning procedure for an ant colony algorithm. In El-Ghazali Talbi, Pierre Liardet, Pierre Collet, Evelyne Lutton and Marc Schoenauer, editors, *Proceedings of the Seventh International Conference on Artificial Evolution*, volume 3871 of *Lecture Notes in Computer Science*, pp. 25–36. Springer. doi:10.1007/11740698_3.
- Allen Chan and Alex A. Freitas (2006b). A new ant colony algorithm for multi-label classification with applications in bioinformatics. In *Proceedings of the Eighth Annual Conference on Genetic and Evolutionary Computation*, pp. 27–34. ACM. doi:10.1145/1143997.1144002.
- Kuo-Chu Chang and Robert Fung (1995). Symbolic probabilistic inference with both discrete and continuous variables. *IEEE Transactions on Systems, Man and Cybernetics*, 25(6):910–916. doi:10.1109/21.384253.

- R. Martin Chavez and Gregory F. Cooper (1990). An empirical evaluation of a randomized algorithm for probabilistic inference. In Max Henrion, Ross Shachter, Laveen Kanal and John Lemmer, editors, *Uncertainty in Artificial Intelligence 5*, pp. 191–208. North-Holland.
- Mark Chavira and Adnan Darwiche (2007). Compiling Bayesian networks using variable elimination. In Manuela M. Veloso, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pp. 2443–2449. Morgan Kaufmann.
- Peter Cheeseman and John Stutz (1996). Bayesian classification (AutoClass): Theory and results. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pp. 153–180. AAAI Press.
- Jian Cheng and Marek Druzdzal (2001). Confidence inference in Bayesian networks. In Jack Breese and Daphne Koller, editors, *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pp. 75–82. Morgan Kaufmann.
- Jian Cheng and Marek J. Druzdzal (2000). AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks. *Journal of Artificial Intelligence Research*, 13:155–188.
- Jie Cheng, David A. Bell and Weiru Liu (1997). An algorithm for Bayesian belief network construction from data. In *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics*, pp. 83–90.
- Jie Cheng, Russell Greiner, Jonathan Kelly, David Bell and Weiru Liu (2002). Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137(1–2):43–90. doi:10.1016/S0004-3702(02)00191-1.
- David Chickering and David Heckerman (1999). Fast learning from sparse data. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 109–115. Morgan Kaufmann.
- David Chickering, David Heckerman and Christopher Meek (1997a). A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pp. 80–89. Morgan Kaufmann.

- David Chickering, David Heckerman and Christopher Meek (1997b). A Bayesian approach to learning Bayesian networks with local structure. Technical Report MSR-TR-97-07, Microsoft Research.
- David Chickering and Christopher Meek (2002). Finding optimal Bayesian networks. In *Proceedings of the Eighteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pp. 94–102. Morgan Kaufmann, San Francisco, CA.
- David M. Chickering (1996a). Learning Bayesian networks is NP-complete. In D. Fisher and H. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, chapter 12, pp. 121–130. Springer.
- David M. Chickering (1996b). Learning equivalence classes of Bayesian network structures. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 150–157. Morgan Kaufmann.
- David M. Chickering (2002a). Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498.
- David Maxwell Chickering (1995). A transformational characterization of equivalent Bayesian network structures. In Philippe Besnard and Steve Hanks, editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pp. 87–98. Morgan Kaufmann.
- David Maxwell Chickering (2002b). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554. doi:10.1162/153244303321897717.
- David Maxwell Chickering, D. Geiger and D. Heckerman (1995). Learning Bayesian networks: Search methods and experimental results. In *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, pp. 112–128.
- David Maxwell Chickering and David Heckerman (1997). Efficient approximations for the marginal likelihood of Bayesian networks with hidden variables. *Machine Learning*, 29(2–3):181–212. doi:10.1023/A:1007469629108.
- David Maxwell Chickering, David Heckerman and Christopher Meek (2004). Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330.

- David Maxwell Chickering and Christopher Meek (2006). On the incompatibility of faithfulness and monotone DAG faithfulness. *Artificial Intelligence*, 170(8–9):653–666. doi:10.1016/j.artint.2006.03.001.
- C. K. Chow and C. N. Liu (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.
- Peter Clark and Tim Niblett (1989). The CN2 induction algorithm. *Machine Learning*, 3(4):261–283. doi:10.1023/A:1022641700528.
- W. J. Conover (1999). *Practical Nonparametric Statistics*. John Wiley & Sons, Third edition.
- Gregory F. Cooper (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2–3):393–405. doi:10.1016/0004-3702(90)90060-D.
- Gregory F. Cooper (1995). A Bayesian method for learning belief networks that contain hidden variables. *Journal of Intelligent Information Systems*, 4(1):71–88. doi:10.1007/BF00962823.
- Gregory F. Cooper (1997). A simple constraint-based algorithm for efficiently mining observational databases for causal relationships. *Data Mining and Knowledge Discovery*, 1(2):203–224. doi:10.1023/A:1009787925236.
- Gregory F. Cooper and Edward Herskovits (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347. doi:10.1007/BF00994110.
- Gregory F. Cooper and Changwon Yoo (1999). Causal discovery from a mixture of experimental and observational data. In Henri Prade and Kathryn Laskey, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 116–125. Morgan Kaufmann.
- Elon S. Correa, Alex A. Freitas and Colin G. Johnson (2007). Particle swarm and Bayesian networks applied to attribute selection for protein functional classification. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 2651–2658. ACM. doi:10.1145/1274000.1274081.

- D. Costa and A. Hertz (1997). Ants can colour graphs. *Journal of the Operational Research Society*, 48(3):295–305. doi:10.1057/palgrave.jors.2600357.
- Carlos Cotta and Jorge Muruzábal (2002). Towards a more efficient evolutionary induction of Bayesian networks. In *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII)*, volume 2439 of *Lecture Notes in Computer Science*, pp. 730–739. Springer. doi:10.1007/3-540-45712-7_70.
- Carlos Cotta and Jorge Muruzábal (2004). On the learning of Bayesian network graph structures via evolutionary programming. In Peter Lucas, editor, *Proceedings of the Second European Workshop on Probabilistic Graphical Models*, pp. 65–72.
- Steve B. Cousins, William Chena and Mark E. Frisse (1993). A tutorial introduction to stochastic simulation algorithms for belief networks. *Artificial Intelligence in Medicine*, 5(4):315–340. doi:10.1016/0933-3657(93)90020-4.
- Robert Cowell (2001). Conditions under which conditional independence and scoring methods lead to identical selection of Bayesian network models. In Jack Breese and Daphne Koller, editors, *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pp. 91–97. Morgan Kaufmann.
- Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen and David J. Spiegelhalter (1999). *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer.
- Nicandro Cruz-Ramírez, Héctor-Gabriel Acosta-Mesa, Rocío-Erandi Barrientos-Martínez and Luis-Alonso Nava-Fernández (2006). How good are the Bayesian information criterion and the minimum description length principle for selection? A Bayesian network analysis. In *Proceedings of the Fifth Mexican International Conference on Artificial Intelligence (MICA 2006)*, volume 4293 of *Lecture Notes in Artificial Intelligence*, pp. 494–504. Springer. doi:10.1007/11925231_46.
- Paul Dagum, Adam Galper and Eric Horvitz (1992). Dynamic network models for forecasting. In Didier Dubois, Michael P. Wellman, Bruce D'Ambrosio and Philippe Smets, editors, *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence (UAI-92)*, pp. 41–48. Morgan Kaufmann.
- Paul Dagum and Eric Horvitz (1993). A bayesian analysis of simulation algorithms for inference in belief networks. *Networks*, 23(5):499–516. doi:10.1002/net.3230230506.

- Paul Dagum and Michael Luby (1993). Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–154. doi:10.1016/0004-3702(93)90036-B.
- Paul Dagum and Michael Luby (1997). An optimal approximation algorithm for Bayesian inference. *Artificial Intelligence*, 93(1–2):1–27. doi:10.1016/S0004-3702(97)00013-1.
- Rónán Daly and Qiang Shen (2007). Methods to accelerate the learning of Bayesian network structures. In George M. Coghill, editor, *Proceedings of the 2007 UK Workshop on Computational Intelligence*.
- Rónán Daly and Qiang Shen (2008). Learning Bayesian network equivalence classes with ant colony optimization. *Revised version for journal publication*.
- Rónán Daly, Qiang Shen and Stuart Aitken (2006a). Speeding up the learning of equivalence classes of Bayesian network structures. In A. P. del Pobil, editor, *Proceedings of the Tenth IASTED International Conference on Artificial Intelligence and Soft Computing*, pp. 34–39. ACTA Press.
- Rónán Daly, Qiang Shen and Stuart Aitken (2006b). Using ant colony optimization in learning Bayesian network equivalence classes. In Xue Z. Wang and Rui Fa Li, editors, *Proceedings of the 2006 UK Workshop on Computational Intelligence*, pp. 111–118.
- Rónán Daly, Qiang Shen and Stuart Aitken (2008). A review of the literature on learning Bayesian networks. *Under review for journal publication*.
- Adnan Darwiche (1995). Conditioning methods for exact and approximate inference in causal networks. In Philippe Besnard and Steve Hanks, editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pp. 99–107. Morgan Kaufmann.
- Adnan Darwiche (1998). Dynamic jointrees. In Gregory F. Cooper and Serafín Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 97–104. Morgan Kaufmann.
- Adnan Darwiche (2001a). Decomposable negation normal form. *Journal of the ACM*, 48(4):608–647. doi:10.1145/502090.502091.
- Adnan Darwiche (2001b). Recursive conditioning. *Artificial Intelligence*, 126(1–2):5–41. doi:10.1016/S0004-3702(00)00069-2.

- Adnan Darwiche (2002). A logical approach to factoring belief networks. In Dieter Fensel, Fausto Giunchiglia, Deborah L. McGuinness and Mary-Anne Williams, editors, *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR-02)*, pp. 409–420. Morgan Kaufmann.
- Adnan Darwiche (2003). A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3):280–305. doi:10.1145/765568.765570.
- Sanjoy Dasgupta (1997). The sample complexity of learning fixed-structure Bayesian networks. *Machine Learning*, 29(2–3):165–180. doi:10.1023/A:1007417612269.
- Sanjoy Dasgupta (1999). Learning polytrees. In Henri Prade and Kathryn Laskey, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 134–141. Morgan Kaufmann.
- Denver Dash and Gregory F. Cooper (2004). Model averaging for prediction with discrete Bayesian networks. *Journal of Machine Learning Research*, 5:1177–1203.
- Denver Dash and Marek Druzdzel (2003). A robust independence test for constraint-based learning of causal structure. In Christopher Meek and Uffe Kjærulff, editors, *Proceedings of the Nineteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pp. 167–174. Morgan Kaufmann, San Francisco, CA.
- Denver Dash and Marek J. Druzdzel (1999). A hybrid anytime algorithm for the construction of causal models from sparse data. In Henri Prade and Kathryn Laskey, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 142–149. Morgan Kaufmann.
- Luis M. de Campos (1998). Independency relationships and learning algorithms for singly connected networks. *Journal of Experimental & Theoretical Artificial Intelligence*, 10(4):511–549.
- Luis M. de Campos (2006). A scoring function for learning bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7:2149–2187.
- Luis M. de Campos and Javier G. Castellano (2007). Bayesian network learning algorithms using structural restrictions. *International Journal of Approximate Reasoning*, 45(2):233–254. doi:10.1016/j.ijar.2006.06.009.

- Luis M. de Campos, Juan M. Fernández-Luna, José A. Gámez and José M. Puerta (2002a). Ant colony optimization for learning Bayesian networks. *International Journal of Approximate Reasoning*, 31(3):291–311. doi:10.1016/S0888-613X(02)00091-9.
- Luis M. de Campos, Juan M. Fernández-Luna and J. Miguel Puerta (2003). An iterated local search algorithm for learning Bayesian networks with restarts based on conditional independence tests. *International Journal of Intelligent Systems*, 18(2):221–235. doi:10.1002/int.10085.
- Luis M. de Campos, José A. Gámez and José M. Puerta (2002b). Learning Bayesian networks by ant colony optimisation: Searching in two different spaces. *Mathware & Soft Computing*, 9(2–3).
- Luis M. de Campos and Juan F. Huete (1997). On the use of independence relationships for learning simplified belief networks. *International Journal of Intelligent Systems*, 12(7):495–522. doi:10.1002/(SICI)1098-111X(199707)12:7<495::AID-INT2>3.0.CO;2-G.
- Luis M. de Campos and Juan F. Huete (2000a). Approximating causal orderings for Bayesian networks using genetic algorithms and simulated annealing. In *Proceedings of the Eight Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 333–340.
- Luis M. de Campos and Juan F. Huete (2000b). A new approach for learning belief networks using independence criteria. *International Journal of Approximate Reasoning*, 24(1):11–37. doi:10.1016/S0888-613X(99)00042-0.
- Luis M. de Campos and J. Miguel Puerta (2001a). Stochastic local algorithms for learning belief networks: Searching in the space of the orderings. In Salem Benferhat and Philippe Besnard, editors, *Proceedings of the Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2001)*, volume 2143 of *Lecture Notes in Artificial Intelligence*, pp. 228–239. Springer. doi:10.1007/3-540-44652-4_21.
- Luis M. de Campos and J. Miguel Puerta (2001b). Stochastic local and distributed search algorithms for learning belief networks. In *Proceedings of the Third International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models*, pp. 109–115. ICIMAF, Havana, Cuba.

- Luis Miguel de Campos, Juan Manuel Fernández-Luna and Jose Miguel Puerta (2002c). Local search methods for learning Bayesian networks using a modified neighborhood in the space of DAGs. In *Advances in Artificial Intelligence: Proceedings of the Eight Ibero-American Conference on AI (IBERAMIA 2002)*, volume 2527 of *Lecture Notes in Artificial Intelligence*, pp. 182–192. Springer. doi:10.1007/3-540-36131-6_19.
- Adamo L. de Santana, Carlos R. Frances, Claudio A. Rocha, Solon V. Carvalho, Nandamudi L. Vijaykumar, Liviane P. Rego and Joao C. Costa (2007). Strategies for improving the modeling and interpretability of Bayesian networks. *Data and Knowledge Engineering*, 63(1):91–107. doi:10.1016/j.datak.2006.10.005.
- Thomas Dean and Keiji Kanazawa (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, 5(2):142–150. doi:10.1111/j.1467-8640.1989.tb00324.x.
- Alain Delaplace, Thierry Brouard and Hubert Cardot (2006). Two evolutionary methods for learning Bayesian network structures. In *Proceedings of the International Conference on Computational Intelligence and Security*, volume 1, pp. 137–142. IEEE. doi:10.1109/ICCIAS.2006.294107.
- A. P. Dempster, N. M. Laird and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.
- J. L. Deneubourg, S. Aron, S. Goss and J. M. Pasteels (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2):159–168. doi:10.1007/BF01417909.
- Norbert Dojer (2006). Learning Bayesian networks does not have to be NP-hard. In *Proceedings of the Thirty-First International Symposium on Mathematical Foundations of Computer Science*, volume 4162 of *Lecture Notes in Computer Science*, pp. 305–314. Springer. doi:10.1007/11821069_27.
- Dorit Dor and Michael Tarsi (1992). A simple algorithm to construct a consistent extension of a partially oriented graph. Technical Report R-185, Cognitive Systems Laboratory, Department of Computer Science, UCLA.
- Marco Dorigo (1992). *Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale*. Ph.D. thesis, Politecnico di Milano, Italy.

- Marco Dorigo and Gianni Di Caro (1999). The ant colony optimization meta-heuristic. In David Corne, Marco Dorigo and Fred Glover, editors, *New Ideas in Optimization*, pp. 11–32. McGraw-Hill.
- Marco Dorigo and Luca Maria Gambardella (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66. doi:10.1109/4235.585892.
- Marco Dorigo, Vittorio Maniezzo and Alberto Colorni (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41. doi:10.1109/3477.484436.
- Marco Dorigo and Thomas Stützle (2004). *Ant Colony Optimization*. The MIT Press.
- Denise Draper and Steve Hanks (1994). Localized partial evaluation of belief networks. In Ramon Lopez de Mantaras and David Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pp. 170–177. Morgan Kaufmann.
- Marek Druzdzal and Herbert Simon (1993). Causality in Bayesian belief networks. In David Heckerman and Abe Mamdani, editors, *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence (UAI-93)*, pp. 3–11. Morgan Kaufmann.
- Marek J. Druzdzal (1994). Some properties of joint probability distributions. In Ramon Lopez de Mantaras and David Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pp. 187–194. Morgan Kaufmann.
- Marek J. Druzdzal (1996). Qualitative verbal explanations in Bayesian belief networks. *Artificial Intelligence and Simulation of Behaviour Quarterly*, 94:43–54.
- F. J. Díez (1996). Local conditioning in Bayesian networks. *Artificial Intelligence*, 87(1–2):1–20. doi:10.1016/0004-3702(95)00118-2.
- F. J. Díez and J. Mira (1994). Distributed inference in Bayesian networks. *Cybernetics and Systems*, 25(1):39–61. doi:10.1080/01969729408902314.
- Daniel Eaton and Kevin Murphy (2007a). Bayesian structure learning using dynamic programming and MCMC. In *Proceedings of the Twenty-Third Annual Conference on Uncertainty in Artificial Intelligence (UAI-07)*.

- Daniel Eaton and Kevin Murphy (2007b). Exact Bayesian structure learning from uncertain interventions. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Journal of Machine Learning Research: Workshop and Conference Proceedings*, pp. 107–114.
- Gal Elidan and Nir Friedman (2001). Learning the dimensionality of hidden variables. In *Proceedings of the Seventeenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pp. 144–151. Morgan Kaufmann.
- Gal Elidan, Noam Lotner, Nir Friedman and Daphne Koller (2001). Discovering hidden variables: A structure-based approach. In Todd K. Leen, Thomas G. Dietterich and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pp. 479–485. MIT Press.
- Gal Elidan, Matan Ninio, Nir Friedman and Dale Schuurmans (2002). Data perturbation for escaping local maxima in learning. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, pp. 132–139. AAAI Press.
- Eli Faulkner (2007). K2GA: Heuristically guided evolution of Bayesian network structures from data. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007)*, pp. 18–25. doi:10.1109/CIDM.2007.368847.
- Jeff Forbes, Tim Huang, Keiji Kanazawa and Stuart Russell (1995). The BATmobile: Towards a Bayesian automated taxi. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 95)*, pp. 1878–1885. Morgan Kaufmann.
- Nir Friedman (1997). Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML '97)*, pp. 125–133. Morgan Kaufmann.
- Nir Friedman (1998). The Bayesian structural EM algorithm. In Gregory F. Cooper and Serafín Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 129–138. Morgan Kaufmann.
- Nir Friedman (2004). Inferring cellular networks using probabilistic graphical models. *Science*, 303(5679):799–805. doi:10.1126/science.1094068.
- Nir Friedman, Dan Geiger and Moises Goldszmidt (1997). Bayesian network classifiers. *Machine Learning*, 29(2–3):131–163. doi:10.1023/A:1007465528199.

- Nir Friedman and Lise Getoor (1999). Efficient learning using constrained sufficient statistics. In David Heckerman and Joe Whittaker, editors, *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*. Morgan Kaufmann.
- Nir Friedman and Moises Goldszmidt (1996a). Discretizing continuous attributes while learning Bayesian networks. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML '96)*, pp. 157–165. Morgan Kaufmann.
- Nir Friedman and Moises Goldszmidt (1996b). Learning Bayesian networks with local structure. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 252–262. Morgan Kaufmann.
- Nir Friedman and Moises Goldszmidt (1997). Sequential update of Bayesian network structure. In Dan Geiger and Prakash P. Shenoy, editors, *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pp. 165–174. Morgan Kaufmann.
- Nir Friedman, Moises Goldszmidt and Abraham Wyner (1999a). Data analysis with Bayesian networks: A bootstrap approach. In Henri Prade and Kathryn Laskey, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 196–205. Morgan Kaufmann.
- Nir Friedman, Moises Goldszmidt and Abraham Wyner (1999b). On the application of The Bootstrap for computing confidence measures on features of induced Bayesian networks. In David Heckerman and Joe Whittaker, editors, *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, pp. 197–202. Morgan Kaufmann.
- Nir Friedman and Daphne Koller (2000). Being Bayesian about network structure. In Craig Boutilier and Moises Goldszmidt, editors, *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pp. 201–210. Morgan Kaufmann.
- Nir Friedman and Daphne Koller (2003). Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1–2):95–125. doi:10.1023/A:1020249912095.
- Nir Friedman, Michal Linial, Iftach Nachman and Dana Pe'er (2000). Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3/4):601–620.

- Nir Friedman, Kevin Murphy and Stuart Russell (1998). Learning the structure of dynamic probabilistic networks. In Gregory F. Cooper and Serafin Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 139–148. Morgan Kaufmann.
- Nir Friedman, Iftach Nachman and Dana Pe'er (1999c). Learning Bayesian network structure from massive datasets: The “Sparse Candidate” algorithm. In Henri Prade and Kathryn Laskey, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 206–215. Morgan Kaufmann.
- Nir Friedman and Zohar Yakhini (1996). On the sample complexity of learning Bayesian networks. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 274–282. Morgan Kaufmann.
- Robert Fung and Kuo-Chu Chang (1990). Weighing and integrating evidence for stochastic simulation in Bayesian networks. In Max Henrion, Ross Shachter, Laveen Kanal and John Lemmer, editors, *Uncertainty in Artificial Intelligence 5*, pp. 209–219. North-Holland.
- Robert M. Fung and Stuart L. Crawford (1990). Constructor: A system for the induction of probabilistic models. In *Proceedings of the Eight National Conference on Artificial Intelligence*, volume 2, pp. 762–769. AAAI Press.
- Luca M. Gambardella and Marco Dorigo (1995). Ant-Q: A reinforcement learning approach to the travelling salesman problem. In Armand Prieditis and Stuart J. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning (ICML 1995)*, pp. 252–260. Morgan Kaufmann.
- Luca Maria Gambardella and Marco Dorigo (2000). An ant colony system hybridized with a new local search for the sequential ordering problem. *INFORMS Journal on Computing*, 12(3):237–255. doi:10.1287/ijoc.12.3.237.12636.
- Song Gao, Qinkun Xiao, Quan Pan and Qingguo Li (2007). Learning dynamic bayesian networks structure based on Bayesian optimization algorithm. In *Advances in Neural Networks: Proceedings of the Fourth International Symposium on Neural Networks (ISNN 2007)*, volume 4492 of *Lecture Notes in Computer Science*, pp. 424–431. Springer. doi:10.1007/978-3-540-72393-6_51. Part II.

- Dan Geiger (1998). Graphical models and exponential families. In Gregory F. Cooper and Serafin Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 156–165. Morgan Kaufmann.
- Dan Geiger and David Heckerman (1994). Learning Gaussian networks. Technical Report MSR-TR-94-10, Microsoft Research.
- Dan Geiger and David Heckerman (1995). A characterization of the Dirichlet distribution with application to learning Bayesian networks. In Philippe Besnard and Steve Hanks, editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pp. 196–207. Morgan Kaufmann.
- Dan Geiger and David Heckerman (1997). A characterization of the Dirichlet distribution through global and local parameter independence. *The Annals of Statistics*, 25(3):1344–1369. doi:10.1214/aos/1069362752.
- Dan Geiger, David Heckerman, Henry King and Christopher Meek (2001). Stratified exponential families: Graphical models and model selection. *The Annals of Statistics*, 29(2):505–529. doi:10.1214/aos/1009210550.
- Dan Geiger, David Heckerman and Christopher Meek (1996). Asymptotic model selection for directed networks with hidden variables. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 283–290. Morgan Kaufmann.
- Dan Geiger, Azaria Paz and Judea Pearl (1990). Learning causal trees from dependence information. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI 1990)*, pp. 770–776. AAAI Press.
- Zoubin Ghahramani (1998). Learning dynamic Bayesian networks. In C. Lee Giles and M. Gori, editors, *Adaptive Processing of Sequences and Data Structures*, volume 1387 of *Lecture Notes in Artificial Intelligence*, pp. 168–197. Springer. doi:10.1007/BFb0053999.
- Zoubin Ghahramani and Michael I. Jordan (1997). Factorial hidden Markov models. *Machine Learning*, 29(2–3):245–273. doi:10.1023/A:1007425814087.
- Steven Gillispie and Michael D. Perlman (2001). Enumerating Markov equivalence classes of acyclic digraph models. In Jack Breese and Daphne Koller, editors, *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pp. 171–177. Morgan Kaufmann.

- Steven B. Gillispie and Michael D. Perlman (2002). The size distribution for Markov equivalence classes of acyclic digraph models. *Artificial Intelligence*, 141(1-2):137-155. doi:10.1016/S0004-3702(02)00264-3.
- Paolo Giudici and Robert Castelo (2003). Improving Markov chain Monte Carlo model search for data mining. *Machine Learning*, 50(1-2):127-158. doi:10.1023/A:1020202028934.
- Paolo Giudici, Peter Green and Claudia Tarantola (1999). Efficient model determination for discrete graphical models. Discussion Paper 99-93, Department of Statistics, Athens University of Economics and Business.
- Paolo Giudici and Peter J. Green (1999). Decomposable graphical Gaussian model determination. *Biometrika*, 86(4):785-801. doi:10.1093/biomet/86.4.785.
- Fred Glover (1989). Tabu search—Part I. *ORSA Journal on Computing*, 1(3):190-206.
- Fred Glover (1990). Tabu search—Part II. *ORSA Journal on Computing*, 2(1):4-32.
- Clark Glymour and Gregory F. Cooper, editors (1999). *Computation, Causation, & Discovery*. The MIT Press.
- E. Mark Gold (1967). Language identification in the limit. *Information and Control*, 10(5):447-474. doi:10.1016/S0019-9958(67)91165-5.
- Anna Goldenberg and Andrew Moore (2004). Tractable learning of large Bayes net structures from sparse data. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML '04)*, pp. 44-51. ACM. doi:10.1145/1015330.1015406.
- Kui Xiang Gou, Gong Xiu Jun and Zheng Zhao (2007). Learning Bayesian network structure from distributed homogeneous data. In *Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, volume 3, pp. 250-254. IEEE. doi:10.1109/SNPD.2007.472.
- Russell Greiner, Adam Grove and Dale Schuurmans (1997). Learning Bayesian nets that perform well. In Dan Geiger and Prakash P. Shenoy, editors, *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pp. 198-207. Morgan Kaufmann.

- Haipeng Guo and William Hsu (2002). A survey of algorithms for real-time Bayesian network inference. In Haipeng Guo, Eric Horvitz, William H. Hsu and Eugene Santos Jr., editors, *Papers from the AAAI Workshop on Real-Time Decision Support and Diagnosis Systems*, pp. 1–12. AAAI Press.
- Yuan-Yuan Guo, Man-Leung Wong and Zhi-Hua Cai (2006). A novel hybrid evolutionary algorithm for learning Bayesian networks from incomplete data. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2006)*, pp. 916–923. doi:10.1109/CEC.2006.1688409.
- Elias Gyftodimos and Peter A. Flach (2004). Hierarchical Bayesian networks: An approach to classification and learning for structured data. In *Methods and Applications of Artificial Intelligence: Proceedings of the Third Hellenic Conference on AI (SETN 2004)*, volume 3025 of *Lecture Notes in Artificial Intelligence*, pp. 291–300. Springer. doi:10.1007/b97168.
- José A. Gámez and José M. Puerta (2002). Searching for the best elimination sequence in Bayesian networks by using ant colony optimization. *Pattern Recognition Letters*, 23(1–3):261–277. doi:10.1016/S0167-8655(01)00123-4.
- D. E. Heckerman, E. J. Horvitz and B. N. Nathwani (1992). Toward normative expert systems: Part I. The Pathfinder project. *Methods of Information in Medicine*, 31(2):90–105.
- David Heckerman (1995a). A Bayesian approach to learning causal networks. Technical Report MSR-TR-95-04, Microsoft Research.
- David Heckerman (1995b). A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research.
- David Heckerman (2007). A Bayesian approach to learning causal networks. In Ward Edwards and Ralph F. Miles, Jr., editors, *Advances in Decision Analysis: From Foundations to Applications*, chapter 11, pp. 202–220. Cambridge University Press.
- David Heckerman and John S. Breese (1996). Causal independence for probability assessment and inference using Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics—Part A*, 26(6):826–831. doi:10.1109/3468.541341.
- David Heckerman and Dan Geiger (1995). Likelihoods and parameter priors for Bayesian networks. Technical Report MSR-TR-95-54, Microsoft Research.

- David Heckerman, Dan Geiger and David M. Chickering (1995). Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243. doi:10.1023/A:1022623210503.
- Xing-Chen Heng, Zheng Qin, Xian-Hui Wang and Li-Ping Shao (2006). Research on learning Bayesian networks by particle swarm optimization. *Information Technology Journal*, 5(3):540–545.
- Max Henrion (1988). Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In John F. Lemmer and Laveen N. Kanal, editors, *Uncertainty in Artificial Intelligence 2*, pp. 149–163. North-Holland.
- Luis D. Hernández, Serafin Moral and Antonio Salmerón (1998). A Monte Carlo algorithm for probabilistic propagation in belief networks based on importance sampling and stratified simulation techniques. *International Journal of Approximate Reasoning*, 18(1–2):53–91. doi:10.1016/S0888-613X(97)10004-4.
- Eddie Herskovits and Gregory Cooper (1991). Kutató: An entropy-driven system for construction of probabilistic expert systems from data. In Piero Bonissone, Max Henrion, Laveen Kanal and John Lemmer, editors, *Uncertainty in Artificial Intelligence 6*, pp. 54–62. North-Holland.
- Jennifer A. Hoeting, David Madigan, Adrian E. Raftery and Chris T. Volinsky (1999). Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–417. doi:10.1214/ss/1009212519.
- Reimar Hofmann and Volker Tresp (1996). Discovering structure in continuous variables using Bayesian networks. In David S. Touretzky, Michael C. Mozer and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, volume 8, pp. 500–506. The MIT Press.
- Nicholas Holden and Alex A. Freitas (2004). Web page classification with an ant colony algorithm. In Xin Yao, Edmund Burke, José A. Lozano, Jim Smith, Juan J. Merelo-Guervós, John A. Bullinaria, Jonathan Rowe, Peter Tiño, Ata Kabán and Hans-Paul Schwefel, editors, *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pp. 1092–1102. Springer. doi:10.1007/b100601.

- Nicholas Holden and Alex A. Freitas (2005). A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 100–107. IEEE. doi:10.1109/SIS.2005.1501608.
- Nicholas Holden and Alex A. Freitas (2006). Hierarchical classification of g-protein-coupled receptors with a PSO/ACO algorithm. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 77–84.
- Nicholas Holden and Alex A. Freitas (2007). A hybrid PSO/ACO algorithm for classification. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pp. 2745–2750. ACM. doi:10.1145/1274000.1274080.
- Gary F. Holness (2007). A direct measure for the efficacy of Bayesian network structures learned from data. In *Proceedings of the Fifth International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM 2007)*, volume 4571 of *Lecture Notes in Artificial Intelligence*, pp. 601–615. Springer. doi:10.1007/978-3-540-73499-4_45.
- Wei-Chiang Hong, Yu-Fen Chen, Peng-Wen Chen and Yi-Hsuan Yeh (2007). Continuous ant colony optimization algorithms in a support vector regression based financial forecasting model. In Jingsheng Lei, JingTao Yao and Qingfu Zhang, editors, *Proceedings of the Third International Conference on Natural Computation*, volume 1, pp. 548–552. IEEE. doi:10.1109/ICNC.2007.315.
- William H. Hsu, Haipeng Guo, Benjamin B. Perry and Julie A. Stilson (2002). A permutation genetic algorithm for variable ordering in learning Bayesian networks from data. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*, pp. 383–390. Morgan Kaufmann.
- Cecil Huang and Adnan Darwiche (1996). Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263. doi:10.1016/S0888-613X(96)00069-2.
- Jiejun Huang, Heping Pan and Youchuan Wan (2005). An algorithm for cooperative learning of Bayesian network structure from data. In *Proceedings of the Eight International Conference on Computer Supported Cooperative Work in Design (CSCWD 2004)*, volume 3168 of *Lecture Notes in Computer Science*, pp. 86–94. Springer. doi:10.1007/11568421_9.

- Kurt Huang and Max Henrion (1996). Efficient search-based inference for noisy-OR belief networks: TopEpsilon. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 325–331. Morgan Kaufmann.
- Yimin Huang and Marco Valtorta (2006). Identifiability in causal Bayesian networks: A sound and complete algorithm. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, volume 2, pp. 1149–1154. AAAI Press.
- Zan Huang, Jiexun Lib, Hua Su, George S. Watts and Hsinchun Chen (2007). Large-scale regulatory network analysis from microarray data: modified Bayesian network learning and association rule mining. *Decision Support Systems*, 43(4):1207–1225. doi:10.1016/j.dss.2006.02.002.
- Dirk Husmeier (2003). Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19(17):2271–2282.
- Kyu-Baek Hwang, Byoung-Hee Kim and Byoung-Tak Zhang (2006). Learning hierarchical Bayesian networks for large-scale data analysis. In *Proceedings of the Thirteenth International Conference on Neural Information Processing (ICONIP 2006)*, volume 4232 of *Lecture Notes in Computer Science*, pp. 670–679. Springer. doi:10.1007/11893028_75.
- Kyu-Baek Hwang, Jae Won Lee, Seung-Woo Chung and Byoung-Tak Zhang (2002). Construction of large-scale bayesian networks by local to global search. In *Trends in Artificial Intelligence : Proceedings of the Seventh Pacific Rim International Conference on Artificial Intelligence (PRICAI 2002)*, volume 2417 of *Lecture Notes in Artificial Intelligence*, pp. 375–384. Springer.
- Seiya Imoto, Takao Goto and Satoru Miyano (2002). Estimation of genetic networks and functional structures between genes by using Bayesian networks and nonparametric regression. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter and Teri E. Klein, editors, *Proceedings of the Seventh Pacific Symposium on Biocomputing*, pp. 175–186. World Scientific.
- Seiya Imoto, Tomoyuki Higuchi, Takao Goto, Kousuke Tashiro, Satoru Kuhara and Satoru Miyano (2003). Combining microarrays and biological knowledge for estimating gene networks via Bayesian networks. In *Proceedings of the IEEE Bioinformatics Conference (CSB 2003)*, pp. 104–113. doi:10.1109/CSB.2003.1227309.

- Tommi Jaakkola and Michael I. Jordan (1997). Recursive algorithms for approximating probabilities in graphical models. In Michael Mozer, Michael I. Jordan and Thomas Petsche, editors, *Advances in Neural Information Processing Systems 9 (NIPS*1996)*, pp. 487–493. MIT Press.
- Tommi S. Jaakkola and Michael I. Jordan (1996). Computing upper and lower bounds on likelihoods in intractable networks. A.I. Memo 1571, Artificial Intelligence Lab, Massachusetts Institute of Technology.
- Tommi S. Jaakkola and Michael I. Jordan (1999a). Improving the mean field approximation via the use of mixture distributions. In Michael I. Jordan, editor, *Learning in Graphical Models*, pp. 163–174. MIT Press.
- Tommi S. Jaakkola and Michael I. Jordan (1999b). Variational probabilistic inference and the QMR-DT network. *Journal of Artificial Intelligence Research*, 10:291–322.
- Carlos M. Jarque and Anil K. Bera (1980). Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics Letters*, 6(3):255–259. doi:10.1016/0165-1765(80)90024-5.
- Allan Leck Jensen (1995). *A probabilistic model based decision support system for mildew management in winter wheat*. Ph.D. thesis, Dina Foulum, Research Center Foulum, Aalborg University.
- Finn V. Jensen and Frank Jensen (1994). Optimal junction trees. In Ramon Lopez de Mantaras and David Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pp. 360–366. Morgan Kaufmann.
- Finn V. Jensen, S. L. Lauritzen and K. G. Olesen (1990a). Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282.
- Finn V. Jensen and Thomas D. Nielsen (2007). *Bayesian Networks and Decision Graphs*. Information Science and Statistics. Springer, second edition.
- Finn Verner Jensen, Kristian G. Olesen and Stig Kjaer Andersen (1990b). An algebra of Bayesian belief universes for knowledge-based systems. *Networks*, 20(5):637–659. doi:10.1002/net.3230200509.

- Haiyang Jia, Dayou Liu, Juan Chen and Xin Liu (2007). A hybrid approach for learning Markov equivalence classes of Bayesian network. In *Proceedings of the Second International Conference on Knowledge Science, Engineering and Management (KSEM 2007)*, volume 4798 of *Lecture Notes in Artificial Intelligence*, pp. 611–616. Springer. doi:10.1007/978-3-540-76719-0_67.
- George John and Pat Langley (1995). Estimating continuous distributions in Bayesian classifiers. In Philippe Besnard and Steve Hanks, editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pp. 338–345. Morgan Kaufmann.
- Anders Jonsson and Andrew Barto (2007). Active learning of dynamic Bayesian networks in Markov decision processes. In *Proceedings of the Seventh International Symposium on Abstraction, Reformulation, and Approximation (SARA 2007)*, volume 4612 of *Lecture Notes in Artificial Intelligence*, pp. 273–284. Springer. doi:10.1007/978-3-540-73580-9_22.
- Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola and Lawrence K. Saul (1999). An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233. doi:10.1023/A:1007665907178.
- Markus Kalisch and Peter Bühlmann (2007). Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8:613–636.
- Keiji Kanazawa, Daphne Koller and Stuart Russell (1995). Stochastic simulation algorithms for dynamic probabilistic networks. In Philippe Besnard and Steve Hanks, editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pp. 346–351. Morgan Kaufmann.
- Mehmet Kayaalp and Gregory F. Cooper (2002). A Bayesian network scoring metric that is based on globally uniform parameter priors. In Adnan Darwiche and Nir Friedman, editors, *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pp. 251–258. Morgan Kaufmann.
- James Kennedy and Russell Eberhart (1995). Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, pp. 1942–1948. doi:10.1109/ICNN.1995.488968.

- James Kennedy and Russell C. Eberhart (1997). A discrete binary version of the particle swarm optimization algorithm. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 5, pp. 4104–4108. doi:10.1109/ICSMC.1997.637339.
- Russell J. Kennett, Kevin B. Korb and Ann E. Nicholson (2001). Seabreeze prediction using Bayesian networks. In *Proceedings of the Fifth Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD 2001)*, volume 2035 of *Lecture Notes in Artificial Intelligence*, pp. 148–153. Springer. doi:10.1007/3-540-45357-1_18.
- Jin H. Kim and Judea Pearl (1983). A computational model for causal and diagnostic reasoning in inference systems. In Alan Bundy, editor, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI 83)*, pp. 190–193. William Kaufmann.
- Kyung-Joong Kim and Sung-Bae Cho (2006). Evolutionary aggregation and refinement of Bayesian networks. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2006)*, pp. 1513–1520. doi:10.1109/CEC.2006.1688488.
- Sun Yong Kim, Seiya Imoto and Satoru Miyano (2003). Inferring gene networks from time series microarray data using dynamic Bayesian networks. *Briefings in Bioinformatics*, 4(3):228–235. doi:10.1093/bib/4.3.228.
- S. Kirkpatrick, C. D. Gelatt Jr. and M. P. Vecchi (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680. doi:10.1126/science.220.4598.671.
- Uffe Kjærulff (1992a). A computational scheme for reasoning in dynamic probabilistic networks. In Didier Dubois, Michael P. Wellman, Bruce D'Ambrosio and Philippe Smets, editors, *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence (UAI-92)*, pp. 121–129. Morgan Kaufmann.
- Uffe Kjærulff (1992b). Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, 2(1):7–17. doi:10.1007/BF01890544.
- Uffe Kjærulff (1993). Approximation of Bayesian networks through edge removals. Technical Report IR-93-2007, Department of Mathematics and Computer Science, Aalborg University.
- Uffe Kjærulff (1994). Reduction of computational complexity in Bayesian networks through removal of weak dependences. In Ramon Lopez de Mantaras and David Poole,

- editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pp. 374–382. Morgan Kaufmann.
- Uffe Kjærulff (1997). Nested junction trees. In Dan Geiger and Prakash P. Shenoy, editors, *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pp. 294–301. Morgan Kaufmann.
- Mikko Koivisto (2006). Advances in exact Bayesian structure discovery in Bayesian networks. In *Proceedings of the Twenty-Second Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*. AUAI Press.
- Mikko Koivisto and Kismat Sood (2004). Exact bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573.
- Kevin B. Korb and Ann E. Nicholson (2004). *Bayesian Artificial Intelligence*. Series in Computer Science and Data Analysis. Chapman & Hall/CRC.
- Tomáš Kočka, Remco R. Bouckaert and Milan Studený (2001). On the inclusion problem. Research Report 2010, Institute of Information Theory and Automation, Prague.
- Tomáš Kočka and Robert Castelo (2001). Improved learning of Bayesian networks. In Jack Breese and Daphne Koller, editors, *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pp. 269–276. Morgan Kaufmann.
- Kristian Kristensen and Ilse A. Rasmussen (2002). The use of a Bayesian network in the design of a decision support system for growing malting barley without use of pesticides. *Computers and Electronics in Agriculture*, 33(3):197–217. doi:10.1016/S0168-1699(02)00007-8.
- Harry H. Ku and Solomon Kullback (1969). Approximating discrete probability distributions. *IEEE Transactions on Information Theory*, 15(4):444–447.
- S. Kullback and R. A. Leibler (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86. doi:10.1214/aoms/1177729694.
- Chee-Keong Kwoh and Duncan Fyfe Gillies (1996). Using hidden nodes in Bayesian networks. *Artificial Intelligence*, 88(1–2):1–38. doi:10.1016/0004-3702(95)00119-0.
- Wai Lam (1998). Bayesian network refinement via machine learning approach. *Transactions on Pattern Analysis and Machine Intelligence*, 20(3):240–251. doi:10.1109/34.667882.

- Wai Lam and Faheim Bacchus (1994a). Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10(3):269–293. doi:10.1111/j.1467-8640.1994.tb00166.x.
- Wai Lam and Fahiem Bacchus (1993). Using causal information and local measures to learn Bayesian networks. In David Heckerman and Abe Mamdani, editors, *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence (UAI-93)*, pp. 243–250. Morgan Kaufmann.
- Wai Lam and Fahiem Bacchus (1994b). Using new data to refine a Bayesian network. In Ramon Lopez de Mantaras and David Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pp. 383–390. Morgan Kaufmann.
- Pedro Larrañaga, Cindy M. H. Kuijpers, Roberto H. Murga and Yosu Yurramendi (1996a). Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics—Part A*, 26(4):487–493. doi:10.1109/3468.508827.
- Pedro Larrañaga, Mikel Poza, Yosu Yurramendi, Roberto H. Murga and Cindy M. H. Kuijpers (1996b). Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *Transactions on Pattern Analysis and Machine Intelligence*, 18(9):912–926. doi:10.1109/34.537345.
- S. L. Lauritzen and N. Wermuth (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17(1):31–57. doi:10.1214/aos/1176347003.
- Steffen L. Lauritzen (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis*, 19(2):191–201. doi:10.1016/0167-9473(93)E0056-A.
- Steffen L. Lauritzen and David J. Spiegelhalter (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224.
- Philippe Leray and Olivier François (2005). Bayesian network structural learning and incomplete data. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR 2005)*.

- Arthur M. Lesk (2002). *Introduction to Bioinformatics*. Oxford University Press.
- John Levine and Frederick Ducatelle (2004). Ant colony optimisation and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society*, 55(7):705–716. doi:10.1057/palgrave.jors.2601771.
- Gang Li, Honghua Dai and Yiqing Tu (2002). Linear causal model discovery using the MML criterion. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 2002)*, pp. 274–281. doi:10.1109/ICDM.2002.1183913.
- Xiao-Lin Li, Shuang-Cheng Wang and Xiang-Dong He (2006). Learning Bayesian networks structures based on memory binary particle swarm optimization. In *Proceedings of the Sixth International Conference on Simulated Evolution and Learning (SEAL 2006)*, volume 4247 of *Lecture Notes in Computer Science*, pp. 568–574. Springer. doi:10.1007/11903697_72.
- Zhaoyu Li and Bruce D'Ambrosio (1994). Efficient inference in Bayes networks as a combinatorial optimization problem. *International Journal of Approximate Reasoning*, 11(1):55–81. doi:10.1016/0888-613X(94)90019-1.
- Yan Lin and Marek J. Druzdzal (1999). Stochastic sampling and search in belief updating algorithms for very large Bayesian networks. In *Working Notes of the AAAI Spring Symposium on Search Techniques for Problem Solving Under Uncertainty and Incomplete Information*, pp. 77–82.
- Feng Liu, Fengzhan Tian and Qiliang Zhu (2007a). Bayesian network structure ensemble learning. In *Proceedings of the Third International Conference on Advanced Data Mining and Applications (ADMA 2007)*, volume 4632 of *Lecture Notes in Artificial Intelligence*, pp. 454–465. Springer. doi:10.1007/978-3-540-73871-8_42.
- Feng Liu, Fengzhan Tian and Qiliang Zhu (2007b). An improved greedy Bayesian network learning algorithm on limited data. In *Proceedings of the Seventeenth International Conference on Artificial Neural Networks (ICANN 2007)*, volume 4668 of *Lecture Notes in Computer Science*, pp. 49–57. Springer. doi:10.1007/978-3-540-74690-4_6. Part I.
- Feng Liu and QiLiang Zhu (2007a). The max-relevance and min-redundancy greedy Bayesian network learning algorithm. In *Bio-inspired Modeling of Cognitive Tasks: Proceedings of the Second International Work-Conference on the Interplay Between*

- Natural and Artificial Computation (IWINAC 2007)*, volume 4527 of *Lecture Notes in Computer Science*, pp. 346–356. Springer. doi:10.1007/978-3-540-73053-8_35. Part I.
- Feng Liu and Qiliang Zhu (2007b). Max-relevance and min-redundancy greedy Bayesian network learning on high dimensional data. In *Proceedings of the Third International Conference on Natural Computation (ICNC 2007)*, volume 1, pp. 217–221. doi:10.1109/ICNC.2007.467.
- Yan-Peng Liu, Ming-Guang Wu and Ji-Xin Qian (2006). Evolving neural networks using the hybrid of ant colony optimization and BP algorithms. In Jun Wang, Zhang Yi, Jacek M. Zurada, Bao-Liang Lu and Hujun Yin, editors, *Proceedings of the Third International Symposium on Neural Networks (ISNN 2006)*, volume 3971 of *Lecture Notes in Computer Science*, pp. 714–722. Springer. doi:10.1007/11759966_105.
- Peter Lucas (2002). Restricted Bayesian network structure learning. In Jose A. Gámez and Antonio Salmeron, editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM 2002)*, pp. 117–126.
- Peter J. F. Lucas, Linda C. van der Gaag and Ameen Abu-Hanna (2004). Bayesian networks in biomedicine and health-care. *Artificial Intelligence in Medicine*, 30(3):201–214. doi:10.1016/j.artmed.2003.11.001.
- David Madigan, Steen A. Andersson, Michael D. Perlman and Chris T. Volinsky (1996). Bayesian model averaging and model selection for Markov equivalence classes of acyclic digraphs. *Communications in Statistics - Theory and Methods*, 25(11):2493–2519. doi:10.1080/03610929608831853.
- David Madigan, Jonathan Gavrin and Adrian E. Raftery (1994). Enhancing the predictive performance of Bayesian graphical models. Technical Report 270, Department of Statistics, University of Washington.
- David Madigan, Krzysztof Mosurski and Russell G. Almond (1997). Graphical explanation in belief networks. *Journal of Computational and Graphical Statistics*, 6(2):160–181.
- David Madigan and Adrian E. Raftery (1994). Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, 89(428):1535–1546.

- David Madigan, Adrian E. Raftery, Jeremy C. York, Jeffrey M. Bradshaw and Russell G. Almond (1993). Strategies for graphical model selection. In *Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics*, pp. 331–336.
- David Madigan, Jeremy York and Denis Allard (1995). Bayesian graphical models for discrete data. *International Statistical Review*, 63(2):215–232.
- F.M. Malvestuto (1991). Approximating discrete probability distributions with decomposable models. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(5):1287–1294. doi:10.1109/21.120082.
- Vittorio Maniezzo and Alberto Colorni (1999). The ant system applied to the quadratic assignment problem. *IEEE Transactions on Knowledge and Data Engineering*, 11(5):769–778. doi:10.1109/69.806935.
- H. B. Mann and D. R. Whitney (1947). On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60. doi:10.1214/aoms/1177730491.
- Vikash Mansinghka, Charles Kemp, Thomas Griffiths and Joshua Tenenbaum (2006). Structured priors for structure learning. In *Proceedings of the Twenty-Second Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*. AUAI Press.
- Dimitris Margaritis (2004). Distribution-free learning of graphical model structure in continuous domains. Technical Report TR-ISU-CS-04-06, Department of Computer Science, Iowa State University.
- Dimitris Margaritis and Sebastian Thrun (2000). Bayesian network induction via local neighborhoods. In S. A. Solla, T. K. Leen and K. R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pp. 505–511. MIT Press.
- Massimiliano Mascherini and Federico M. Stefanini (2007). Using weak prior information on structures to learn bayesian networks. In *Proceedings of the Eleventh International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2007)*, volume 4692 of *Lecture Notes in Artificial Intelligence*, pp. 413–420. Springer. doi:10.1007/978-3-540-74819-9_51. Part I.
- C. Robertson McClung (2006). Plant circadian rhythms. *The Plant Cell*, 18:792–803.

- Robert McGill, John W. Tukey and Wayne A. Larsen (1978). Variations of box plots. *The American Statistician*, 32(1):12–16.
- Christopher Meek (1995). Causal inference and causal explanation with background knowledge. In Philippe Besnard and Steve Hanks, editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pp. 403–410. Morgan Kaufmann.
- Christopher Meek and David Heckerman (1997). Structure and parameter learning for causal independence and causal interaction models. In Dan Geiger and Prakash P. Shenoy, editors, *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pp. 366–375. Morgan Kaufmann.
- Marina Meilă and Tommi Jaakkola (2006). Tractable Bayesian learning of tree belief networks. *Statistics and Computing*, 16(1):77–92. doi:10.1007/s11222-006-5535-3.
- Elliot M. Meyerowitz (2001). Prehistory and history of Arabidopsis research. *Plant Physiology*, 125(1):15–19.
- Microsoft Research (1995). Win95pts. A model for printer troubleshooting in Microsoft Windows 95.
- Blackford Middleton, Michael Shwe, David Heckerman, Max Henrion, Eric Horvitz, Harold Lehmann and Gregory Cooper (1991). Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base: II. Evaluation of diagnostic performance. *Methods of Information in Medicine*, 30(4):256–267.
- I. Miguel and Q. Shen (2001). Solution techniques for constraint satisfaction problems: Advanced approaches. *Artificial Intelligence Review*, 15(4):269–293. doi:10.1023/A:1011096320004.
- R. Mondragón-Becerra, N. Cruz-Ramírez, D. A. Garcia-López, K. Gutiérrez-Fragoso, W. A. Luna-Ramírez, G. Ortiz-Hernández and C. A. Piña-García (2006). Automatic construction of bayesian network structures by means of a concurrent search mechanism. In *MICAI 2006: Advances in Artificial Intelligence Proceedings of the Fifth Mexican International Conference on Artificial Intelligence*, volume 4293 of *Lecture Notes in Artificial Intelligence*, pp. 652–662. Springer. doi:10.1007/11925231_62.
- Stefano Monti and Gregory Cooper (1998). A multivariate discretization method for learning Bayesian networks from mixed data. In Gregory F. Cooper and Serafín

Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 404–413. Morgan Kaufmann.

Stefano Monti and Gregory F. Cooper (1996). Bounded recursive decomposition: A search-based method for belief-network inference under limited resources. *International Journal of Approximate Reasoning*, 15(1):49–75. doi:10.1016/0888-613X(96)00012-6.

Stefano Monti and Gregory F. Cooper (1997a). Learning Bayesian belief networks with neural network estimators. In Michael C. Mozer, Michael I. Jordan and Thomas Petsche, editors, *Advances in Neural Information Processing Systems 9*, pp. 578–584. The MIT Press.

Stefano Monti and Gregory F. Cooper (1997b). Learning hybrid Bayesian networks from data. Technical Report ISSP-97-01, Intelligent Systems Program, University of Pittsburgh.

Andrew Moore and Mary Soon Lee (1998). Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91.

Andrew Moore and Weng-Keen Wong (2003). Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In Tom Fawcett and Nina Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning*, pp. 552–559. AAAI Press.

Manuel Martínez Morales, Ramiro Garza Domínguez, Nicandro Cruz Ramírez, Alejandro Guerra Hernández and José Luis Jiménez Andrade (2004). A method based on genetic algorithms and fuzzy logic to induce Bayesian networks. In *Proceedings of the Fifth Mexican International Conference in Computer Science (ENC '04)*, pp. 176–180. IEEE Computer Society. doi:10.1109/ENC.2004.1342603.

Heinz Mühlenbein (1997). The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346. doi:10.1162/evco.1997.5.3.303.

Paul Munteanu and Mohamed Bendou (2001). The EQ framework for learning equivalence classes of Bayesian networks. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pp. 417–424. IEEE Computer Society. doi:10.1109/ICDM.2001.989547.

- Paul Munteanu and Denis Cau (2000). Efficient score-based learning of equivalence classes of Bayesian networks. In Djamel A. Zighed, Henryk J. Komorowski and Jan M. Zytkow, editors, *Proceedings of the Fourth European Conference on the Principles of Data Mining and Knowledge Discovery (PKDD 2000)*, volume 1910 of *Lecture Notes in Computer Science*, pp. 96–105. Springer. doi:10.1007/3-540-45372-5_10.
- K. Murphy and S. Mian (1999). Modelling gene expression data using dynamic Bayesian networks. Technical report, Computer Science Division, University of California, Berkeley.
- Kevin P. Murphy, Yair Weiss and Michael I. Jordan (1999). Loopy belief propagation for approximate inference: An empirical study. In Henri Prade and Kathryn Laskey, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 467–475. Morgan Kaufmann.
- Kevin Patrick Murphy (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, University of California, Berkeley.
- Jorge Muruzábal and Carlos Cotta (2004). A primer on the evolution of equivalence classes of Bayesian-network structures. In Xin Yao, Edmund Burke, José A. Lozano, Jim Smith, Juan J. Merelo-Guervós, John A. Bullinaria, Jonathan Rowe, Peter Tiño, Ata Kabán and Hans-Paul Schwefel, editors, *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature - PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pp. 612–621. Springer. doi:10.1007/b100601.
- James W. Myers, Kathryn B. Laskey and Kenneth A. DeJong (1999a). Learning Bayesian networks from incomplete data using evolutionary algorithms. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pp. 458–465. Morgan Kaufmann.
- James W. Myers, Kathryn Blackmond Laskey and Tod S. Levitt (1999b). Learning Bayesian networks from incomplete data with stochastic search algorithms. In Henri Prade and Kathryn Laskey, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 476–484. Morgan Kaufmann.
- Radford M. Neal (1992). Connectionist learning of belief networks. *Artificial Intelligence*, 56(1):71–113. doi:doi:10.1016/0004-3702(92)90065-6.

- Radford M. Neal and Geoffrey E. Hinton (1999). A view of the EM algorithm that justifies incremental, sparse, and other variants. In Michael I. Jordan, editor, *Learning in Graphical Models*. MIT Press.
- Richard E. Neapolitan (2004). *Learning Bayesian Networks*. Series in Artificial Intelligence. Prentice Hall.
- Julian Neil, Chris Wallace and Kevin Korb (1999). Learning Bayesian networks with restricted causal interactions. In Henri Prade and Kathryn Laskey, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pp. 486–493. Morgan Kaufmann.
- Julian R. Neil and Kevin B. Korb (1999). The evolution of causal models: A comparison of Bayesian metrics and structure priors. In *Proceedings of the Third Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining (PAKDD '99)*, volume 1574 of *Lecture Notes in Artificial Intelligence*, pp. 432–437. Springer. doi:10.1007/3-540-48912-6_57.
- Jens Nielsen, Tomas Kocka and Jose Peña (2003). On local optima in learning Bayesian networks. In Uffe Kjærulff Christopher Meek, editor, *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pp. 435–444. Morgan Kaufmann.
- Andrew J. Novobilski (2003). The random selection and manipulation of legally encoded Bayesian networks in genetic algorithms. In *Proceedings of the First International Conference on Artificial Intelligence (IC-AI '03)*, pp. 438–443. CSREA Press.
- Andreas Nägele, Mathäus Dejori and Martin Stetter (2007). Bayesian substructure learning - approximate learning of very large network structures. In *Proceedings of the Eighteenth European Conference on Machine Learning (ECML 2007)*, volume 4701 of *Lecture Notes in Artificial Intelligence*, pp. 238–249. Springer. doi:10.1007/978-3-540-74958-5_24.
- R. T. O'Donnell, A. E. Nicholson, B. Han, K. B. Korb, M. J. Alam and L. R. Hope (2006a). Causal discovery with prior information. In *Proceedings of the Nineteenth Australian Joint Conference on Artificial Intelligence (AI 2006)*, volume 4303 of *Lecture Notes in Artificial Intelligence*, pp. 1162–1167. Springer. doi:10.1007/11941439_141.
- R.T. O'Donnell, L. Allison and K.B. Korb (2006b). Learning hybrid Bayesian networks by MML. In *Advances in Artificial Intelligence: Proceedings of the Nineteenth Australian*

- Joint Conference on Artificial Intelligence (AI 2006)*, volume 4304 of *Lecture Notes in Artificial Intelligence*, pp. 192–203. Springer. doi:10.1007/11941439_23.
- Sascha Ott, S. Imoto and Satoru Miyano (2004). Finding optimal models for small gene networks. In *Proceedings of the Ninth Pacific Symposium on Biocomputing*, pp. 557–567. World Scientific.
- Sascha Ott and Satoru Miyano (2003). Finding optimal gene networks using biological constraints. *Genome Informatics*, 14:124–133.
- Payam Pakzad and Venkat Anantharam (2002). Belief propagation and statistical physics. In *Proceedings of the 2000 Conference on Information Sciences and Systems*.
- James D. Park and Adnan Darwiche (2004). A differential semantics for jointree algorithms. *Artificial Intelligence*, 156(2):197–216. doi:10.1016/j.artint.2003.04.004.
- Rafael S. Parpinelli, Heitor S. Lopes and Alex A. Freitas (2001). An ant colony based system for data mining: Applications to medical data. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 791–798. Morgan Kaufmann.
- Rafael S. Parpinelli, Heitor S. Lopes and Alex A. Freitas (2002a). An ant colony algorithm for classification rule discovery. In H. A. Abbass, R. A. Sarker and C. S. Newton, editors, *Data Mining: A Heuristic Approach*, pp. 191–208. Idea Group Publishing, London.
- Rafael S. Parpinelli, Heitor S. Lopes and Alex A. Freitas (2002b). Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 6(4):321–332. doi:10.1109/TEVC.2002.802452.
- Rafael S. Parpinelli, Heitor S. Lopes and Alex A. Freitas (2002c). Mining comprehensible rules from data with an ant colony algorithm. In Guilherme Bittencourt and Geber L. Ramalho, editors, *Proceedings of the Sixteenth Brazilian Symposium on Artificial Intelligence*, volume 2507 of *Lecture Notes in Artificial Intelligence*, pp. 259–269. Springer. doi:10.1007/3-540-36127-8_25.
- Judea Pearl (1982). Reverend Bayes on inference engines: A distributed hierarchical approach. In David L. Waltz, editor, *Proceedings of the Second National Conference on Artificial Intelligence*, pp. 133–136. The AAAI Press.
- Judea Pearl (1986a). A constraint - propagation approach to probabilistic reasoning. In Laveen N. Kanal and John F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pp. 357–369. North-Holland.

- Judea Pearl (1986b). Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288. doi:10.1016/0004-3702(86)90072-X.
- Judea Pearl (1987). Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32(2):245–257. doi:10.1016/0004-3702(87)90012-9.
- Judea Pearl (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Series in Representation and Reasoning. Morgan Kaufmann.
- Judea Pearl (2000). *Causality*. Cambridge University Press.
- Judea Pearl and Tom S. Verma (1991). A theory of inferred causation. In James F. Allen, Richard Fikes and Erik Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pp. 441–452. Morgan Kaufmann, San Mateo, California.
- Karl Pearson (1896). Mathematical contributions to the theory of evolution.—III. Regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 187:253–318.
- Hanchuan Peng and Chris Ding (2003). Structure search and stability enhancement of Bayesian networks. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003)*, pp. 621–624. doi:10.1109/ICDM.2003.1250992.
- Mark A. Peot and Ross D. Shachter (1991). Fusion and propagation with multiple observations in belief networks. *Artificial Intelligence*, 48(3):3. doi:10.1016/0004-3702(91)90030-N.
- Michael D. Perlman (2001). Graphical model search via essential graphs. Technical Report 367, Department of Statistics, University of Washington.
- Bruno-Edouard Perrin, Liva Ralaivola, Aurélien Mazurie, Samuele Bottani, Jacques Mallet and Florence d’Alché Buc (2003). Gene networks inference using dynamic Bayesian networks. *Bioinformatics*, 19 Suppl. 2:ii138–ii148.
- D.T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim and M. Zaidi (2006). The bees algorithm — A novel tool for complex optimisation problems. In *Proceedings of the Intelligent Production Machines and Systems Conference*, pp. 454–461.
- David Poole (1993a). Average-case analysis of a search algorithm for estimating prior and posterior probabilities in Bayesian networks with extreme probabilities. In Ruzena

- Bajcsy, editor, *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI 93)*, pp. 606–612. Morgan Kaufmann.
- David Poole (1993b). The use of conflicts in searching Bayesian networks. In David Heckerman and Abe Mamdani, editors, *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence (UAI-93)*, pp. 359–367. Morgan Kaufmann.
- David Poole (1996). Probabilistic conflicts in a search algorithm for estimating posterior probabilities in Bayesian networks. *Artificial Intelligence*, 88(1–2):69–100. doi:10.1016/S0004-3702(96)00022-7.
- David Poole (1997). Probabilistic partial evaluation: Exploiting rule structure in probabilistic inference. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI 97)*, pp. 1284–1291. Morgan Kaufmann.
- David Poole (1998). Context-specific approximation in probabilistic inference. In Gregory F. Cooper and Serafín Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pp. 447–454. Morgan Kaufmann.
- David Poole and Nevin Lianwen Zhang (2003). Exploiting contextual independence in probabilistic inference. *Journal of Artificial Intelligence Research*, 18:263–313.
- Malcolm Pradhan and Paul Dagum (1996). Optimal Monte-Carlo estimation of belief network inference. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 446–453. Morgan Kaufmann.
- Gregory M. Provan and Moninder Singh (1996). Learning Bayesian networks using feature selection. In *Learning from Data: Artificial Intelligence and Statistics V*, volume 112 of *Lecture Notes in Statistics*, pp. 291–. Springer.
- J. R. Quinlan (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.
- Marco Ramoni and Paola Sebastiani (1997a). Learning Bayesian networks from incomplete databases. Technical Report KMI-TR-43, Knowledge Media Institute, The Open University.
- Marco Ramoni and Paola Sebastiani (1997b). The use of exogenous knowledge to learn Bayesian networks from incomplete databases. In *Proceedings of the Second International Symposium on Advances in Intelligent Data Analysis, Reasoning about*

- Data (IDA '97)*, volume 1280 of *Lecture Notes in Computer Science*, pp. 537–548. Springer. doi:10.1007/BFb0052869.
- Marco Ramoni and Paola Sebastiani (1999). Learning conditional probabilities from incomplete databases: An experimental comparison. In David Heckerman and Joe Whittaker, editors, *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*. Morgan Kaufmann.
- Marco Ramoni and Paola Sebastiani (2001). Robust learning with missing data. *Machine Learning*, 45(2):147–170. doi:10.1023/A:1010968702992.
- Lene Kolind Rasmussen (1995). *Bayesian Network for Blood Typing and Parentage Verification of Cattle*. Ph.D. thesis, Dina Foulum, Research Center Foulum.
- George Rebane and Judea Pearl (1987). The recovery of causal poly-trees from statistical data. In Laveen N. Kanal, Tod S. Levitt and John F. Lemmer, editors, *Uncertainty in Artificial Intelligence 3*, pp. 175–182. North-Holland.
- Carsten Riggelsen and Ad Feelders (2005). Learning Bayesian network models from incomplete data using importance sampling. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pp. 301–308. Society for Artificial Intelligence and Statistics.
- J. Rissanen (1978). Modeling by shortest data description. *Automatica*, 14(5):465–471. doi:10.1016/0005-1098(78)90005-5.
- Thomas A. Runkler (2005). Ant colony optimization of clustering models. *International Journal of Intelligent Systems*, 20(12):1233–1251. doi:10.1002/int.20111.
- Stuart J. Russell, John Binder, Daphne Koller and Keiji Kanazawa (1995). Local learning in probabilistic networks with hidden variables. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 95)*, volume 2, pp. 1146–1152. Morgan Kaufmann.
- Ferat Sahin and Archana Devasia (2007). Distributed particle swarm optimization for structural Bayesian network learning. In Felix T. S. Chan and Manoj Kumar Tiwari, editors, *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, chapter 27, pp. 505–532. I-Tech Education and Publishing, Vienna, Austria.

- Eugene Santos, Jr., Solomon Shimony and Edward Williams (1996). Sample-and-accumulate algorithms for belief updating in Bayes networks. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 477–484. Morgan Kaufmann.
- Eugene Santos, Jr. and Solomon Eyal Shimony (1998). Deterministic approximation of marginal probabilities in Bayes nets. *IEEE Transactions on Systems, Man, and Cybernetics—Part A*, 28(4):377–393. doi:10.1109/3468.686701.
- Eugene Santos, Jr., Solomon Eyal Shimony and Edward Williams (1997). Hybrid algorithms for approximate belief updating in Bayes nets. *International Journal of Approximate Reasoning*, 17(2–3):191–216. doi:10.1016/S0888-613X(97)00012-1.
- Sumit Sarkar and Ishwar Murthy (1996). Constructing efficient belief network structures with expert provided information. *IEEE Transactions on Knowledge and Data Engineering*, 8(1):134–143. doi:10.1109/69.485642.
- F. E. Satterthwaite (1946). An approximate distribution of estimates of variance components. *Biometrics Bulletin*, 2(6):110–114.
- Tuija Schmidt and Prakash P. Shenoy (1998). Some improvements to the Shenoy-Shafer and Hugin architectures for computing marginals. *Artificial Intelligence*, 102(2):323–333. doi:10.1016/S0004-3702(98)00047-2.
- Oliver Schulte, Wei Luo and Russell Greiner (2007). Mind change optimal learning of Bayes net structure. In *Learning Theory: Proceedings of the Twentieth Annual Conference on Learning Theory (COLT 2007)*, volume 4539 of *Lecture Notes in Artificial Intelligence*, pp. 187–202. Springer. doi:10.1007/978-3-540-72927-3_15.
- Gideon Schwarz (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464. doi:10.1214/aos/1176344136.
- Ross Shachter, Stig Andersen and Peter Szolovits (1994). Global conditioning for probabilistic inference in belief networks. In Ramon Lopez de Mantaras and David Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pp. 514–522. Morgan Kaufmann.
- Ross Shachter and Mark Peot (1990). Simulation approaches to general probabilistic inference on belief networks. In Max Henrion, Ross Shachter, Laveen Kanal and John Lemmer, editors, *Uncertainty in Artificial Intelligence 5*, pp. 221–234. North-Holland.

- Ross D. Shachter (1986a). Evaluating influence diagrams. *Operations Research*, 34(6):871–882.
- Ross D. Shachter (1986b). Intelligent probabilistic inference. In Laveen N. Kanal and John F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pp. 371–382. North-Holland.
- Ross D. Shachter (1988). Probabilistic inference and influence diagrams. *Operations Research*, 36(4):589–604.
- Glenn R. Shafer and Prakash P. Shenoy (1990). Probability propagation. *Annals of Mathematics and Artificial Intelligence*, 2(1–4):327–351. doi:10.1007/BF01531015.
- Patrick Shaughnessy and Gary Livingston (2005). Evaluating the causal explanatory value of Bayesian network structure learning algorithms. Research Paper 2005-013, Department of Computer Science, University of Massachusetts Lowell.
- Prakash Shenoy and Glenn Shafer (1990). Axioms for probability and belief-function propagation. In *Readings in Uncertain Reasoning*, chapter 7, pp. 575–610. Morgan Kaufmann.
- Prakash P. Shenoy (1997). Binary join trees for computing marginals in the Shenoy-Shafer architecture. *International Journal of Approximate Reasoning*, 17(2–3):239–263. doi:10.1016/S0888-613X(97)89135-9.
- Solomon Eyal Shimony and Eugene Santos, Jr. (1996). Exploiting case-based independence for approximating marginal probabilities. *International Journal of Approximate Reasoning*, 14(1):25–54. doi:10.1016/0888-613X(95)00112-T.
- Michael Shwe and Gregory Cooper (1991). An empirical analysis of likelihood-weighting simulation on a large, multiply connected medical belief network. *Computers and Biomedical Research*, 24(5):453–475. doi:10.1016/0010-4809(91)90020-W.
- Michael Shwe, Blackford Middleton, David Heckerman, Max Henrion, Eric Horvitz, Harold Lehmann and Gregory Cooper (1991). Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base: I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30(4):241–255.
- Tomi Silander, Petri Kontkanen and Petri Myllymaki (2007). On sensitivity of the MAP Bayesian network structure to the equivalent sample size parameter. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI-07)*.

- Tomi Silander and Petri Myllymaki (2006). A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the Twenty-Second Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*. AUAI Press.
- Ajit P. Singh and Andrew W. Moore (2005). Finding optimal Bayesian networks by dynamic programming. Technical Report CMU-CALD-05-106, School of Computer Science, Carnegie Mellon University.
- Moninder Singh and Marco Valtorta (1993). An algorithm for the construction of Bayesian network structures from data. In David Heckerman and Abe Mamdani, editors, *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence (UAI-93)*, pp. 259–265. Morgan Kaufmann.
- Moninder Singh and Marco Valtorta (1995). Construction of Bayesian network structures from data: A brief survey and an efficient algorithm. *International Journal of Approximate Reasoning*, 12(2):111–131. doi:10.1016/0888-613X(94)00016-V.
- James Smaldon and Alex A. Freitas (2006). A new version of the ant-miner algorithm discovering unordered rule sets. In *Proceedings of the Eighth Conference on Genetic and Evolutionary Computation*, pp. 43–50. ACM. doi:10.1145/1143997.1144004.
- Padhraic Smyth (1997). Belief networks, hidden Markov models, and Markov random fields: A unifying view. *Pattern Recognition Letters*, 18(11–13):1261–1268. doi:10.1016/S0167-8655(97)01050-7.
- David J. Spiegelhalter (1986). Probabilistic reasoning in predictive expert systems. In Laveen N. Kanal and John F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pp. 47–67. North-Holland.
- David J. Spiegelhalter and Steffen L. Lauritzen (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20(5):579–605.
- Peter Spirtes and Clark Glymour (1990). An algorithm for fast recovery of sparse causal graphs. Report CMU-PHIL-15, Department of Philosophy, Carnegie Mellon University.
- Peter Spirtes, Clark Glymour and Richard Scheines (2000). *Causation, Prediction, and Search*. Adaptive Computation and Machine Learning. The MIT Press, second edition.

- Peter Spirtes and Chris Meek (1995). Learning Bayesian networks with discrete variables from data. In *Proceedings of First International Conference on Knowledge Discovery and Data Mining*, pp. 294–299. Morgan Kaufmann.
- Peter Spirtes, Christopher Meek and Thomas Richardson (1995). Causal inference in the presence of latent variables and selection bias. In Philippe Besnard and Steve Hanks, editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pp. 499–506. Morgan Kaufmann.
- Sampath Srinivas (1993). A generalization of the noisy-or model. In David Heckerman and Abe Mamdani, editors, *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence (UAI-93)*, pp. 208–218. Morgan Kaufmann.
- Harald Steck (2000). On the use of skeletons when learning in Bayesian networks. In Craig Boutilier and Moises Goldszmidt, editors, *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-00)*, pp. 558–565. Morgan Kaufmann.
- Harald Steck and Tommi Jaakkola (2002). On the Dirichlet prior and Bayesian regularization. In *Advances in Neural Information Processing Systems 15*, pp. 697–704. MIT Press.
- Harald Steck and Tommi S. Jaakkola (2003). (Semi-)predictive discretization during model selection. AI Memo 2003-002, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.
- B. Steinsky (2003). Efficient coding of labeled directed acyclic graphs. *Soft Computing*, 7(5):350–356. doi:10.1007/s00500-002-0223-5.
- Student (1908). The probable error of a mean. *Biometrika*, 6:1–25. doi:10.1093/biomet/6.1.1.
- Thomas Stützle (1998). An ant approach to the flow shop problem. In *Proceedings of the Sixth European Congress on Intelligent Techniques and Soft Computing (EUFIT '98)*, volume 3, pp. 1560–1564. ELITE Foundation, Aachen, Germany.
- Thomas Stützle and Holger H. Hoos (2000). *MAX-MIN* ant system. *Future Generation Computer Systems*, 16(8):889–914.
- H. J. Suermondt and G. F. Cooper (1988). Updating probabilities in multiply-connected belief networks. Technical Report SMI-88-0207, Medical Computer Science Group, Stanford University.

- H. Jacques Suermondt and Gregory F. Cooper (1990). Probabilistic inference in multiply connected belief networks using loop cutsets. *International Journal of Approximate Reasoning*, 4(4):283–306. doi:10.1016/0888-613X(90)90003-K.
- H. Jacques Suermondt and Gregory F. Cooper (1991). Initialization for the method of conditioning in Bayesian belief networks. *Artificial Intelligence*, 50(1):83–94. doi:10.1016/0004-3702(91)90091-W.
- Richard S. Sutton and Andrew G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press.
- Joe Suzuki (1993). A construction of Bayesian networks from databases based on an MDL principle. In David Heckerman and Abe Mamdani, editors, *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence (UAI-93)*, pp. 266–273. Morgan Kaufmann.
- Joe Suzuki (1999). Learning Bayesian belief networks based on the MDL principle: An efficient algorithm using the branch and bound technique. *IEICE Transactions on Information and Systems*, E82-D(2):356–367.
- Marc Teyssier and Daphne Koller (2005). Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pp. 584–590. AUAI Press.
- The Arabidopsis Genome Initiative (2000). Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*. *Nature*, 408(6814):796–815. doi:10.1038/35048692.
- Bo Thiesson (1995). Accelerated quantification of Bayesian networks with incomplete data. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pp. 306–311. AAAI Press.
- Bo Thiesson (1997). Score and information for recursive exponential models with incomplete data. In Dan Geiger and Prakash P. Shenoy, editors, *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pp. 453–463. Morgan Kaufmann.
- Bo Thiesson, Christopher Meek, David Maxwell Chickering and David Heckerman (1998a). Learning mixtures of Bayesian networks. Technical Report MSR-TR-97-30, Microsoft Research.

- Bo Thiesson, Christopher Meek, David Maxwell Chickering and David Heckerman (1998b). Learning mixtures of DAG models. Technical Report MSR-TR-97-30, Microsoft Research.
- Fengzhan Tian, Haisheng Li, Zhihai Wang and Jian Yu (2007). Learning Bayesian networks based on a mutual information scoring function and EMI method. In *Advances in Neural Networks: Proceedings of the Fourth International Symposium on Neural Networks (ISNN 2007)*, volume 4492 of *Lecture Notes in Computer Science*, pp. 414–423. Springer. doi:10.1007/978-3-540-72393-6_50. Part II.
- Fengzhan Tian, Hongwei Zhang and Yuchang Lu (2003). Learning Bayesian networks from incomplete data based on EMI method. In *Proceedings of the Third IEEE Conference on Data Mining (ICDM 2003)*, pp. 323–330. doi:10.1109/ICDM.2003.1250936.
- Fengzhan Tian, Hongwei Zhang, Yuchang Lu and Chunyi Shi (2001). Incremental learning of Bayesian networks with hidden variables. In *Proceedings of the First IEEE International Conference on Data Mining (ICDM 2001)*, pp. 651–652. doi:10.1109/ICDM.2001.989594.
- Simon Tong and Daphne Koller (2000). Active learning for parameter estimation in Bayesian networks. In *Advances in Neural Information Processing Systems 13 (NIPS*2000)*, pp. 647–653. MIT Press.
- Simon Tong and Daphne Koller (2001). Active learning for structure in Bayesian networks. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 01)*, pp. 863–869. Morgan Kaufmann.
- Ioannis Tsamardinos, Constantin F. Aliferis, and Alexander Statnikov (2003a). Algorithms for large scale Markov blanket discovery. In Ingrid Russell and Susan M. Haller, editors, *Proceedings of the Sixteenth International FLAIRS Conference*, pp. 376–381. AAAI Press.
- Ioannis Tsamardinos, Constantin F. Aliferis and Alexander Statnikov (2003b). Time and sample efficient discovery of Markov blankets and direct causal relations. In *Proceedings Of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, pp. 673–678. ACM. doi:10.1145/956750.956838.
- Ioannis Tsamardinos, Constantin F. Aliferis, Alexander Statnikov and Laura E. Brown (2003c). Scaling-up Bayesian network learning to thousands of variables using lo-

cal learning techniques. Technical Report DSL-03-02, Department of Biomedical Informatics, Vanderbilt University, Nashville, Tennessee.

Ioannis Tsamardinos, Laura E. Brown and Constantin F. Aliferis (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78. doi:10.1007/s10994-006-6889-7.

Allan Tucker and Xiaohui Liu (1999). Extending evolutionary programming methods to the learning of dynamic Bayesian networks. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pp. 923–929. Morgan Kaufmann.

Allan Tucker and Xiaohui Liu (2004). Learning dynamic Bayesian networks from multivariate time series with changing dependencies. In *Advances in Intelligent Data Analysis V: Proceedings of the Fifth International Symposium on Intelligent Data Analysis (IDA 2003)*, volume 2810 of *Lecture Notes in Computer Science*, pp. 100–110. Springer. doi:10.1007/b13240.

Allan Tucker, Xiaohui Liu and Andrew Ogden-Swift (2001). Evolutionary learning of dynamic probabilistic models with large time lags. *International Journal of Intelligent Systems*, 16(5):621–646. doi:10.1002/int.1027.

Peter van der Putten and Maarten van Someren (2004). A bias-variance analysis of a real world learning problem: The CoIL challenge 2000. *Machine Learning*, 57(1-2):177–195. doi:10.1023/B:MACH.0000035476.95130.99.

Steven van Dijk and Dirk Thierens (2004). On the use of a non-redundant encoding for learning Bayesian networks from data with a GA. In Xin Yao et al., editors, *Proceedings of the Eight International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pp. 141–150. Springer. doi:10.1007/b100601.

Steven van Dijk, Dirk Thierens and Linda C. van der Gaag (2003a). Building a GA from design principles for learning Bayesian networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2723 of *Lecture Notes in Computer Science*, pp. 886–897. Springer. doi:10.1007/3-540-45105-6_101. Part I.

- Steven van Dijk, Linda C. van der Gaag and Dirk Thierens (2003b). A skeleton-based approach to learning Bayesian networks from data. In Nada Lavrač, Dragan Gamberger, Ljupčo Todorovski and Hendrik Blockeel, editors, *Proceedings of the Seventh European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2003)*, volume 2838 of *Lecture Notes in Artificial Intelligence*, pp. 132–143. Springer. doi:10.1007/b13634.
- Robert A. van Engelen (1997). Approximating Bayesian belief networks by arc removal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):916–920. doi:10.1109/34.608295.
- Thomas Verma and Judea Pearl (1991). Equivalence and synthesis of causal models. In Piero Bonissone, Max Henrion, Laveen Kanal and John Lemmer, editors, *Uncertainty in Artificial Intelligence 6*, pp. 255–268. North-Holland.
- Thomas Verma and Judea Pearl (1992). An algorithm for deciding if a set of observed independencies has a causal explanation. In Didier Dubois, Michael P. Wellman, Bruce D'Ambrosio and Philippe Smets, editors, *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence (UAI-92)*, pp. 323–330. Morgan Kaufmann.
- Chris Wallace and David Boulton (1968). An information measure for classification. *The Computer Journal*, 11(2):185–194.
- Chris S. Wallace and Kevin B. Korb (1999). Learning linear causal models by MML sampling. In A. Gammerman, editor, *Causal Models and Intelligent Data Management*, pp. 89–111. Springer.
- Chris S. Wallace, Kevin B. Korb and Honghua Dai (1996). Causal discovery via MML. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML '96)*, pp. 516–524. Morgan Kaufmann.
- Hao Wang, Kui Yu and Hongliang Yao (2006). Learning dynamic Bayesian networks using evolutionary MCMC. In *Proceedings of the International Conference on Computational Intelligence and Security*, volume 1, pp. 45–50. IEEE. doi:10.1109/ICCIAS.2006.294088.
- Mingyi Wang, Zuo Zhou Chen and S. Cloutier (2007). A hybrid Bayesian network learning method for constructing gene networks. *Computational Biology and Chemistry*, 31(5–6):361–372. doi:10.1016/j.compbiolchem.2007.08.005.

- Yair Weiss (2000). Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12(1):1–41. doi:10.1162/089976600300015880.
- B. L. Welch (1947). The generalization of 'Student's' problem when several different population variances are involved. *Biometrika*, 34(1–2):28–35. doi:10.1093/biomet/34.1-2.28.
- Michael P. Wellman and Chao-Lin Liu (1994). State-space abstraction for anytime evaluation of probabilistic networks. In Ramon Lopez de Mantaras and David Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pp. 567–574. Morgan Kaufmann.
- Joe Whittaker (1990). *Graphical Models in Applied Multivariate Statistics*. Wiley.
- Frank Wilcoxon (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.
- Man Leung Wong, Wai Lam and Kwong Sak Leung (1999). Using evolutionary programming and minimum description length principle for data mining of Bayesian networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(2):174–178. doi:10.1109/34.748825.
- Man Leung Wong, Shing Yan Lee and Kwong Sak Leung (2002). A hybrid approach to discover Bayesian networks from databases using evolutionary programming. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*, pp. 498–505. doi:10.1109/ICDM.2002.1183994.
- Man Leung Wong and Kwong Sak Leung (2004). An efficient data mining method for learning Bayesian networks using an evolutionary algorithm-based hybrid approach. *IEEE Transactions on Evolutionary Computation*, 8(4):378–404. doi:10.1109/TEVC.2004.830334.
- Y. Xiang and T. Chu (1999). Parallel learning of belief networks in large and difficult domains. *Data Mining and Knowledge Discovery*, 3(3):315–339. doi:10.1023/A:1009888910252.
- Yang Xiang, S. K. M. Wong and N. Cercone (1996). Critical remarks on single link search in learning belief networks. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 564–571. Morgan Kaufmann.

- Zhengzheng Xing and Dan Wu (2006). Modeling multiple time units delayed gene regulatory network using dynamic Bayesian network. In *Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)*, pp. 190–195. doi:10.1109/ICDMW.2006.120.
- Heng Xing-Chen, Qin Zheng; Tian Lei and Shao Li-Ping (2007a). Learning bayesian network structures with discrete particle swarm optimization algorithm. In *Proceedings of the IEEE Symposium on Foundations of Computational Intelligence (FOCI 2007)*, pp. 47–52. doi:10.1109/FOCI.2007.372146.
- Heng Xing-Chen, Qin Zheng, Tian Lei and Shao Li-Ping (2007b). Research on structure learning of dynamic Bayesian networks by particle swarm optimization. In *Proceedings of the IEEE Symposium on Artificial Life (ALIFE '07)*, pp. 85–91. doi:10.1109/ALIFE.2007.367782.
- Zhong Yan and Chunwei Yuan (2004). Ant colony optimization for feature selection in face recognition. In David Zhang and Anil K. Jain, editors, *Proceedings of the First International on Conference Biometric Authentication (ICBA 2004)*, volume 3072 of *Lecture Notes in Computer Science*, pp. 221–226. Springer. doi:10.1007/b98225.
- Jonathan S. Yedidia, William T. Freeman and Yair Weiss (2001). Generalized belief propagation. In Todd K. Leen, Thomas G. Dietterich and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13 (NIPS*2000)*, pp. 689–695. MIT Press.
- Raanan Yehezkel and Boaz Lerner (2006). Bayesian network structure learning by recursive autonomy identification. In *Structural, Syntactic, and Statistical Pattern Recognition: Proceedings of the Joint IAPR International Workshops, SSPR 2006 and SPR 2006*, volume 4109 of *Lecture Notes in Computer Science*, pp. 154–162. Springer. doi:10.1007/11815921_16.
- Kui Yu, Hao Wang and Xindong Wu (2007). A parallel algorithm for learning Bayesian networks. In *Proceedings of the Eleventh Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD 2007)*, volume 4426 of *Lecture Notes in Artificial Intelligence*, pp. 1055–1063. Springer. doi:10.1007/978-3-540-71701-0_119.
- Nevin Lianwen Zhang (1996). Irrelevance and parameter learning in Bayesian networks. *Artificial Intelligence*, 88(1–2):359–373. doi:10.1016/S0004-3702(96)00035-5.

- Nevin Lianwen Zhang and David Poole (1994a). Intercausal independence and heterogeneous factorization. In Ramon Lopez de Mantaras and David Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence (UAI-94)*, pp. 606–614. Morgan Kaufmann.
- Nevin Lianwen Zhang and David Poole (1994b). A simple approach to Bayesian network computations. In *Proceedings of the Tenth Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 171–178.
- Nevin Lianwen Zhang and David Poole (1996). Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328.
- Nevin Lianwen Zhang and Li Yan (1998). Independence of causal influence and clique tree propagation. *International Journal of Approximate Reasoning*, 19(3–4):335–349. doi:10.1016/S0888-613X(98)10014-2.
- Min Zou and Suzanne D. Conzen (2005). A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21(1):71–79. doi:10.1093/bioinformatics/bth463.
- Or Zuk, Shiri Margel and Eytan Domany (2006). On the number of samples needed to learn the correct structure of a Bayesian network. In *Proceedings of the Twenty-Second Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*. AUAI Press.